REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	02 Sep 97	

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATIONS PROBLEMS	
6. AUTHOR(S)	
Marvin A. Arostegui, Jr.	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
University of Houston	97-020D

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
DEPARTMENT OF THE AIR FORCE AFIT/CIA 2950 P STREET WPAFB OH 45433-7765	

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
DISTRIBUTION STATEMENT A Approved for public release Distribution Unlimited	

13. ABSTRACT *(Maximum 200 words)*

19970909 136

14. SUBJECT TERMS	15. NUMBER OF PAGES
	443
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

ABSTRACT

AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATION PROBLEMS

A Dissertation

Presented to

the Faculty of the College of Business Administration

University of Houston

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

By

Marvin A. Arostegui, Jr.

August 1997

ABSTRACT

AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATION PROBLEMS

Marvin A. Arostegui, Jr.

Operations managers are typically faced with the need to find good solutions to difficult problems. Such problems include job scheduling, assembly line balancing, process layout, project scheduling, and facilities locations. Although optimal solutions are preferable, the combinatorial nature of these problems means that in many cases problems found in practical applications cannot be solved to optimality within reasonable resources. In these cases, operations managers turns to heuristics. Since the early 1980s, much interest has been devoted to the development and application of three general heuristic algorithms: tabu search, simulated annealing, and genetic algorithms. Each of them specifies a strategy for searching the solution space of a problem looking for "good" local optima. From a practical point of view, we would like to know if any of these methods is indeed better than the other two.

In this research study we conduct an empirical comparison of these three heuristic algorithms using three variants of the facilities location problem: capacitated (CFLP), multiple-periods (MP-FLP), and multiple-commodities (MC-FLP). The selection of three different problem structures allowed us to explore the behavior of the heuristics under different circumstances and constraints. Furthermore, none of the heuristics have been previously applied to these problems.

Our empirical work consisted of several stages including the proper selection of problem solutions representations, parameter searches, and benchmarking. Our final stage was a statistical study to compare the performance of each heuristic along two dimensions: same computation time allowed and same amount of information allowed (i.e., number of solutions evaluated). We also compared the performance without any such restrictions.

When all heuristics are limited to the same amount of computation time, tabu search performs consistently well for all three problem types. However, for MP-FLP, the performance of simulated annealing shows no statistically significant difference from tabu search; and, for MC-FLP, the performance of the genetic algorithm shows no statistically significant difference form tabu search. When all heuristics are limited to the same number of solutions they may evaluate, each heuristic does best for one type of problem. Simulated annealing does best for C-FLP, genetic algorithms do best for MP-FLP, and tabu search does best for MC-FLP. When no constraints are place on the heuristics, tabu search does best for CFLP and MC-FLP and simulated annealing does best for MP-FLP.

Overall, tabu search tends to give the most robust results closely followed by simulated annealing. Genetic algorithms do not generally perform well for these types of problems. They do appear to complement the other two heuristics to the extent that genetic algorithms are better suited for problems where a neighborhood search cannot be easily defined or a large number of solutions cannot be efficiently evaluated.

REFERENCES

Adenso-Díaz, Belarmino (1996) An SA/TS Mixture Algorithm for the Scheduling Tardiness Problem. *European Journal of Operational Research*, **88**, 516-524.

Akinc, Umit (1973) A Branch And Bound Procedure for Solving Warehouse Location Problems with Capacity Constraints. *Unpublished Ph.D. Dissertation,* University of Houston.

Akinc, Umit and B.M. Khumawala (1977) An Efficient Branch and Bound Algorithm for the Capacitated Warehouse Location Problem. *Management Science*, **23**(6), 585-594.

Balakrishnan, P. V. and Varghese S. Jacob (1996) Genetic Algorithms for Product Design. *Management Science*, **42**(8), 1105-1117.

Barnes, J. Wesley and Manuel Laguna (1993) Solving the Multiple-Machine Weighted Flow Time Problem Using Tabu Search. *IIE Transactions,* 25(2), 121-128.

Barnes, J. Wesley and John B. Chambers (1995) Solving the Job Shop Scheduling Problem with Tabu Search. *IIE Transactions,* **27,** 257-263.

Ben-Daya, M., and M. Al-Fawzan (1996) A Simulated Annealing Appraoch for the One-Machine Mean Tardiness Scheduling Problem. *European Journal of Operational Research,* **93**, 61-67.

Bulgak, A. A., P. D. Diwan, and B. Inozu (1995) Buffer Size Optimization in Asynchronous Assembly Systems Using Genetic Algorithms. *Computers and Industrial Engineering,* **28**(2), 309-322.

Černý, V. (1985) Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications,* **45**, 41-51.

Chan, K. C. and H. Tansri (1994) A Study of Genetic Crossover Operations on the Facilities Layout Problem. *Computers and Industrial Engineering,* **26**(3), 537-550.

Cheh, K. M., J. B. Goldberg and R. G. Askin (1991) A Note on the Effect of Neighborhood Structure in Simulated Annealing. *Computers and Operations Research*, **18,** 537-547.

Chen, W.-H., and B. Srivastava (1994) Simulated Annealing Procedures for Forming Machine Cells in Group Technology. *European Journal of Operational Research,* **75,** 100-111.

Cheng, Runwei, Mitsuo Gen, and Yasuhiro Tsujimura (1996) A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms—I. Representation. *Computers and Industrial Engineering,* **30**(4), 983-997.

Collins, N. E., R. W. Eglese and B. L. Golden (1988) Simulated Annealing—An Annotated Bibliography. *American Journal of Mathematical Management Science,* **3**(3), 209-307.

Connolly D. (1992) General Purpose Simulated Annealing. *Journal of the Operational Research Society,* **43,** 495-505.

Crainic, Teodor G., Michel Gendreau, Patrick Soriano, and Michel Toulouse (1993) A Tabu Search Procedure for Multi-commodity Location/Allocation with Balancing Requirements. *Annals of Operations Research,* **41,** 359-383.

Davis P.S. and T.L. Ray (1969) A Branch-Bound Algorithm for the Capacitated Facilities Location Problem. *Naval Research Logistics Quarterly.* **16,** 331-344.

Dowsland, Kathryn A. (1996) Genetic Algorithms—a Tool for OR? *Journal of the Operational Research Society,* **47,** 550-561.

Dowsland, Kathryn A. (1993) Some Experiments with Simulated Annealing Techniques for Packing Problems. *European Journal of Operational Research,* **68,** 389-399.

Dudewicz, E.J. and S.R. Dalal (1975) Allocation of Observations in Ranking and Selection with unequal Variances. *Sankhya*, **B37**, 28-78.

Eglese, R. W. (1990) Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research,* **46,** 271-281.

Ellwein, L.B. and P. Gray (1971) Solving Fixed Charge Location-Allocation Problems with Capacity and Configuration Constraints. *AIIE Transactions*, **3**, 290-299.

França, Paulo M., Michel Gendreau, Gilbert Laporte, and Felipe M. Müller (1996) A Tabu Search Heuristic for the Multiprocessor Scheduling Problem With Sequence Dependent Setup Times. *International Journal of Production Economics*, **43**, 79-89.

Garavelli, A.C., O.G. Okogbaa, and V. Nicola (1996) Global Manufacturing Systems: A Model Supported by Genetic Algorithms to Optimize Production Planning. *Computers & Industrial Engineering*, **31**, 193-196.

Gendreau, Michel, Alain Hertz, and Gilbert Laporte (1994) A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science,* **40**(10), 1276-1290.

Geoffrion, A.M. and G.W. Graves (1974) Multi-commodity Distribution Systems Design. *Management Science*, **20**, 822-844.

Glover, F. (1977) Heuristic for Integer Programming Using Surrogate Constraints. *Decision Sciences,* **8,** 156-166.

Glover, F. (1989) Tabu search—Part I. *ORSA Journal on Computing,* **1**(3), 190-206.

Glover, F. (1990a) Tabu Search—Part II. *ORSA Journal on Computing,* **2**(1), 4-32.

Glover, F. (1990b) Tabu Search: A Tutorial. *Interfaces,* **20**(4), 74-94.

Glover, F. (1993) A User's Guide to Tabu Search. *Annals of Operations Research.*

Glover, Fred and Harvey J. Greenberg (1989) New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intlligence. *European Journal of Operational Research,* **39,** 119-130.

Goldberg, David E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Massachusetts: Addison Wesley Publishing Company, Inc.

Golden, B.L. and W.R. Stewart (1985) Empirical Analysis of Heuristics. In *The Travelling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., Chichester: John Wiley & Sons, 207-249.

Hajek, B. (1988) Cooling Schedules for Optimal Annealing. *Mathematics and Operations Research,* **13**(2), 311-329.

Hao, Qi, Zihou Yang, Dingwei Wang, and Zheng Li (1996) Common Due-Date Determination and Sequencing Using Tabu Search. *Computers and Operations Research*, **23**(5), 409-417.

He, Zesheng, Taeyong Yang, and Andy Tiger (1996) An Exchange Heuristic Imbedded with Simulated Annealing for Due-Dates Job-Shop Scheduling. *European Journal of Operational Research,* **91**, 99-117.

Hindi, K. S. (1995) Solving the Single-Item, Capacitated Dynamic Lot-Sizing Problem with Startup and Reservation Costs by Tabu Search. *Computers and Industrial Engineering,* **28**(4), 701-707.

Holland, John H. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor: The university of Michigan Press.

Hormozi, A.M. (1987) An Improved Multi-Period Facility Location Problem. *Unpublished Ph.D. Dissertation*, University of Houston.

Hormozi, A.M. and B.M. Khumawala (1996) An Improved Multi-Period Facility Location Problem. *IIE Transactions*, **28**, 105-114.

Houck, Christopher R., Jeffrey A. Joines and Michael G. Kay (1996) Comparison of Genetic Algorithms, Random Restart and Two-Opt Switching for Solving Large Location-Allocation Problems. *Computers and Operations Research*, **23**(6) 587-596.

Icmeli, Oya and S. Selcuk Erenguc (1994) A Tabu Search Procedure for the Resource Constrained Project Scheduling Problem With Discounted Cash Flows. *Computers and Operations Research*, **21**(8), 841-853.

Ishibuchi, Hisao, Shinta Misaki, and Hideo Tanaka (1995) Modified Simulated Annealing Algorithms for the Flow Show Sequencing Problem. *European Journal of Operational Research,* **81,** 388-398.

Joines, Jeffrey A., C. Thomas Culbreth, and Russell E. King (1996) Manufacturing Cell Design: An Integer Programming Model Employing Genetic Algorithms. *IIE Transactions,* **28**, 69-85.

Khumawala, B.M. (1974) An Efficient Heuristic Procedure for the Capacitated Warehouse Location Problem. *Naval Research Logistics Quarterly*, **21**(4), 609-623.

Kim, Jung-Ug and Yeong-Dae Kim (1996) Simulated Annealing and Genetic Algorithms for Scheduling Products With Multi-Level Product Structure. *Computers and Operations Research*, **23**(9), 857-868.

Kim, Yeo Keun, Yong Ju Kim, and Yeongho Kim (1996) Genetic Algorithms for Assembly Line Balancing With Various Objectives. *Computers and Industrial Engineering,* **30**(3), 397-409.

Kim, Yeong-Dae, Hyeong-Gyu Lim, and Moon_Won Park (1996) Search Heuristics for a Flowshop Scheduling Problem in a Printed Circuit Board Assembly Process. *European Journal of Operational Research*, **91**, 124-143.

Kincaid, Rex K. (1992) Good Solutions to Discrete Noxious Location Problems Via Metaheuristics. *Annals of Operations Research,* **40,** 265-281.

Kirkpatrick S., C. Gelatt, and P. Vecchi (1983) Optimization by Simulated Annealing. *Science,* **220,** 671-679.

Koulamas, C., S. R. Antony, and R. Jaen (1994) A Survey of Simulated Annealing Applications to Operations Research Problems. *Omega, International Journal of Management Science,* **22**(1), 41-56.

Kuehn Alfred A. and Michael J. Hamburger (1963) A Heuristic Program for Locating Warehouses. *Management Science*, **9**(4), 643-666.

Kuik, Roelof, Marc Salomon, Luk N. Van Wassenhove, and Johan Maes (1993) Linear Programming, Simulated Annealing and Tabu Search Heuristics for Lotsizing in Bottleneck Assembly Systems. *IIE Transactions,* 25(1), 62-72.

Laguna, Manuel, James P. Kelly, Jose Luis Gonzalez-Velarde, and Fred Glover (1995) Tabu Search for the Multilevel Generalized Assignment Problem. *European Journal of Operational Research,* **82,** 176-189.

Law, Averill M. and W. David Kelton (1991) *Simulation Modeling & Analysis*, 2nd ed., McGraw-Hill, Inc.

Lee, Jae-Kwan and Yeong-Dae Kim (1996) Search Heuristics for Resource Constrained Project Scheduling. *Journal of the Operational Research Society,* **47,** 678-689.

Leu Yow-Yuh, Lance A. Matheson, and Loren Paul Rees (1994) Assembly Line Balancing Using Genetic Algorithms With Heuristic-Generated Initial Populations and Multiple Evaluation Criteria. *Decision Sciences,* 25(4), 581-606.

Leu, Yow-Yuh, Lance A. Matheson, and Loren Paul Rees (1996) Sequencing Mixed-Model Assembly Lines With Genetic Algorithms. *Computers and Industrial Engineering,* **30**(4), 1027-1036.

Liu, Chih-Ming, Ruey-Li Kao, and An-Hsiang Wang (1994) Solving Location-Allocation Problems with Rectilinear Distances by Simulated Annealing. *Journal of the Operational Research Society,* **45**(11), 1304-1315.

Lundy, M. and A. Mees (1986) Convergence of an Annealing Algorithm. *Mathematical Programming,* **34,** 111-124.

Malmborg, Charles J. (1996) A Genetic Algorithm for Service Level Based Vehicle Scheduling. *European Journal of Operational Research,* **93**, 191-134.

Marett, Richard and Mike Wright (1996) A Comparison of Neighborhood Search Techniques for Multi-Objective Combinatorial Problems. *Computers and Operations Research*, **23**(5), 465-483.

Metropolis, N. A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953) Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics,* **21,** 1087-1091.

Murata, Tadahiko, Hisao Ishibuchi, and Hideo Tanaka (1996) Genetic Algorithms for Flowshop Scheduling Problems. *Computers and Industrial Engineering,* **30**(4), 1061-1071.

Neebe, A.W. and B.M. Khumawala (1981) An Improved Algorithm for the Multi-Commodity Location Problem. *Journal of Operational Research Society*, **32**, 143-149.

Nowicki, Eugeniusz, and Czeslaw Smutnicki (1996) A Fast Tabu Search Algorithm for the Permutation Flow Shop Problem. *European Journal of Operational Research,* **91**, 160-175.

Nowicki, E. and S. Zdrazalka (1996) Single Machine Scheduling With Major and Minor Setup Times: A Tabu Search Approach. *Journal of the Operational Research Society,* **47**, 1054-1064.

Osman, I. H. and C. N. Potts (1989) Simulated Annealing for Permutation Flow-Shop Scheduling. *Omega, International Journal of Management Science,* **17**(6), 551-557.

Reeves, Colin R., ed. (1993) *Modern Heuristic Techniques for Combinatorial Problems.* New York: John Wiley & Sons.

Renaud, Jacques, Gilbert Laporte, and Gayez F. Boctor (1996) A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem. *Computers and Operations Research*, **23**(3), 229-235.

Roach, Anthony and Rakesh Nagi (1996) A Hybrid GA-SA Algorithm for Just-In-Time Scheduling of Multi-Level Assemblies. *Computers and Industrial Engineering*, **30**(4), 1047-1060.

Rolland, Erik, David A. Schilling, and John R. Current (1996) An Efficient Tabu Search Procedure for the p-Median Problem. *European Journal of Operational Research,* **96,** 329-342.

Roodman, G.M. and L.B. Schwarz (1975) Optimal and Heuristic Facility Phase-out Strategies. *AIEE Transactions*, **7**, 177-184.

Roodman, G.M. and L.B. Schwarz (1977) Extension of the Multi-Period Facility Phase-Out Model; New Procedures and Applications to a Phase-in/Phase-out Problem. *AIEE Transactions*, **9**, 103-107.

Sá., G. (1969) Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem. *Operations Research,* **17**, 1005-1016.

Sakawa, Masatoshi, Kosuke Kato, and Tetsuya Mori (1996) Flexible Scheduling in a Machining Center Through Genetic Algorithms. *Computers and Industrial Engineering,* **30**(4), 931-940.

Schmidt, Linda C and John Jackman (1995) Evaluating Assembly Sequences for Automatic Assembly Systems. *IIE Transactions,* **27,** 23-31.

Sinclair, Marius (1993) Comparison of the Performance of Modern Heuristics for Combinatorial Optimization on Real Data. *Computers and Operations Research*, **20**(7), 687-695.

Skorin-Kapov, Darko and Jadranka Skorin-Kapov (1994) On Tabu Search for the Location of Interacting Hub Facilities. *European Journal of Operational Research,* **73,** 502-509.

Sweeney, D.J. and Ronald L. Tatham (1976) An Improved Long Run Model for Multiple Warehouse Location. *Management Science*, **22**(7), 748-758.

Taillard, E. (1991) Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing,* **17,** 433-455.

Taillard, Éric D., Gilbert Laporte, and Michel Gendreau (1996) Vehicle Routeing with Multiple Use of Vehicles. *Journal of the Operational Research Society,* **47**, 1065-1070.

Thompson, Gary M. (1996) A Simulated Annealing Heuristic for Shift Scheduling Using Non-Continuously Available Employees. *Computers and Operations Research*, **23**(3), 275-288.

Van Laarhoven, Peter J. M., and E. H. L. Aarts (1987) *Simulated Annealing: Theory and Applications.* Reidel, Dordrecht.

Van Laarhoven, Peter J. M., Emile H. L. Aarts, and Jan Karel Lenstra (1992) Job Shop Scheduling by Simulated Annealing. *Operations Research,* **40**(1), 113-125.

Vignaux, G. A., and Z. Michalewicz (1991) A Genetic Algorithm for the Linear Transportation Problem. *IEEE Transactions on Systems, Man, and Cybernetics,* **21**(2), 445-452.

Warszawski, A. (1987) Multi-Dimensional Location Problems. *Operational Research Quarterly*, **24**, 165-179.

Wilhelm, Mickey R. and Thomas L. Ward (1987) Solving Quadratic Assignment Problems by Simulated Annealing. *IIE Transactions*, **107**, 107-119.

AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATION PROBLEMS

A Dissertation

Presented to

the Faculty of the College of Business Administration

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Marvin A. Arostegui, Jr.

August 1997

AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATION PROBLEMS

APPROVED:

Basheer M. Khumawala, Professor of Decision and Information Sciences
Chairperson of Committee

Everette S. Gardner, Jr., Professor of Decision and Information Sciences

Sukran N. Kadipasaoglu, Assistant Professor of Decision and Information Sciences

Julio L. Peixoto, Associate Professor of Decision and Information Sciences

Sara M. Freedman, Dean
College of Business Administration

ACKNOWLEDGEMENTS

I would like to express deepest gratitude to Professor Basheer M. Khumawala, my Dissertation Committee Chairman, Ph.D. advisor, and overall mentor. Over the space of three years, he was a constant source of encouragement, advise, and ideas. He provided me all the opportunities I needed to gain the proper experience so I could achieve my goals and engendered a great sense of self confidence by his willingness to trust in me. I would like to thank Professor Sukran N. Kadipasaoglu for her support and interest in my professional development and my research. Such caring attitudes make this process much more enjoyable. My thanks also to Professors Everette S. Gardner, Jr., and Julio L. Peixoto for passing some of their knowledge and experience down to me as well as for their ideas, suggestions, and participation during this research project.

Finally I want to thank the many members of my family whose consistent confidence in me and constant encouragement ensured the last three years were filled only with high and positive moments.

ABSTRACT

AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATION PROBLEMS

A Dissertation

Presented to

the Faculty of the College of Business Administration

University of Houston

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

By

Marvin A. Arostegui, Jr.

August 1997

ABSTRACT

AN EMPIRICAL COMPARISON OF TABU SEARCH, SIMULATED ANNEALING, AND GENETIC ALGORITHMS FOR FACILITIES LOCATION PROBLEMS

Marvin A. Arostegui, Jr.

Operations managers are typically faced with the need to find good solutions to difficult problems. Such problems include job scheduling, assembly line balancing, process layout, project scheduling, and facilities locations. Although optimal solutions are preferable, the combinatorial nature of these problems means that in many cases problems found in practical applications cannot be solved to optimality within reasonable resources. In these cases, operations managers turns to heuristics. Since the early 1980s, much interest has been devoted to the development and application of three general heuristic algorithms: tabu search, simulated annealing, and genetic algorithms. Each of them specifies a strategy for searching the solution space of a problem looking for "good" local optima. From a practical point of view, we would like to know if any of these methods is indeed better than the other two.

In this research study we conduct an empirical comparison of these three heuristic algorithms using three variants of the facilities location problem: capacitated (CFLP), multiple-periods (MP-FLP), and multiple-commodities (MC-FLP). The selection of three different problem structures allowed us to explore the behavior of the heuristics under different circumstances and constraints. Furthermore, none of the heuristics have been previously applied to these problems.

Our empirical work consisted of several stages including the proper selection of problem solutions representations, parameter searches, and benchmarking. Our final stage was a statistical study to compare the performance of each heuristic along two dimensions: same computation time allowed and same amount of information allowed (i.e., number of solutions evaluated). We also compared the performance without any such restrictions.

When all heuristics are limited to the same amount of computation time, tabu search performs consistently well for all three problem types. However, for MP-FLP, the performance of simulated annealing shows no statistically significant difference from tabu search; and, for MC-FLP, the performance of the genetic algorithm shows no statistically significant difference form tabu search. When all heuristics are limited to the same number of solutions they may evaluate, each heuristic does best for one type of problem. Simulated annealing does best for C-FLP, genetic algorithms do best for MP-FLP, and tabu search does best for MC-FLP. When no constraints are place on the heuristics, tabu search does best for CFLP and MC-FLP and simulated annealing does best for MP-FLP.

Overall, tabu search tends to give the most robust results closely followed by simulated annealing. Genetic algorithms do not generally perform well for these types of problems. They do appear to complement the other two heuristics to the extent that genetic algorithms are better suited for problems where a neighborhood search cannot be easily defined or a large number of solutions cannot be efficiently evaluated.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Decision makers in any field have a natural desire to find optimal solutions to their problems. Unfortunately, there is a class of common business operations problems that are extremely difficult to solve optimally. Such problems include job scheduling, assembly line balancing, process layout, project scheduling, and facilities location. Although optimal algorithms have been developed to solve some of these problems, they suffer from combinatorial explosion, where the time required to solve them grows exponentially with the size of the problem. These problems belong to the class of combinatorial problems known as NP-hard, for which no known optimal algorithms exist that can execute in deterministic polynomial time. In most cases, the time and computing resources required to solve practical sized instances become prohibitive.

Nonetheless, decision makers still have a need for sensible solutions. In these cases, heuristic methods are the only viable alternative. Heuristics are approximation algorithms that yield "good" solutions within a reasonable amount of computation time. Van Laarhoven and Aarts (1987) classify heuristics into tailored and general algorithms. Tailored algorithms have a limited applicability to a specific problem. General algorithms are widely applicable to various forms

of combinatorial optimization problems. In fact, general algorithms define a strategy for obtaining approximate solutions to these difficult problems.

1.1 Background

Since the early 1980s, much interest has been devoted to the development and application of three general heuristic algorithms: tabu search, simulated annealing, and genetic algorithms. Each of them specifies a strategy for searching the solution space of a problem looking for "good" local optima (which may be the global optimum or close to it). Tabu search and simulated annealing are based on a neighborhood search and each specifies a strategy to climb out of a local optimum to ensure a wide search of the solution space. Genetic algorithms use an evolutionary, survival of the fittest strategy where new solutions (children) are generated by combining portions of two prior solutions (parents). In the last 20 years or so, researchers have used these general heuristic algorithms to solve a wide array of combinatorial problems. In all this effort, however, it is still not clear if one of these heuristics is consistently better than the other two.

The number of papers published and the variety of subjects addressed as recently as 1996 show the great degree of interest in these heuristics. Tabu search was used to solve problems in scheduling (Franca, et al., 1996; Nowicki and Zdrzalka, 1996; Nowicki and Smutnicki, 1996; Hao, et al., 1996), vehicle routing (Taillard, Laporte, and Gendreau, 1996; Renaud, Laporte, and Boctor, 1996), lot-sizing (Hindi, 1996), and the p-median location problem (Rolland, Schilling, and Current, 1996). Simulated annealing was used extensively to

solve various forms of the scheduling problem (Roach and Nagi, 1996; Ben-Daya and Al-Fawzan, 1996; He, Yang, and Tiger, 1996; Adenso-Diaz, 1996; Thompson, 1996). Finally, genetic algorithms were used in production assignment (Garavelli, Okogbaa, and Violante, 1996), product design (Balakrishnan and Jacob, 1996), scheduling (Leu, Matheson, and Rees, 1996; Cheng, Gen, and Tsujimura, 1996; Sakawa, Kato, and Mori, 1996), assembly line balancing (Kim, Kim, and Kim, 1996), and manufacturing cell design (Joines, Culbreth, and King, 1996). In most cases, empirical studies show the new heuristics are superior or at least as good as existing, problem specific heuristics.

The broad interest and applicability of these heuristics leads to the natural question of which is better. Some papers have been published comparing tabu search versus simulated annealing (Kim, Lim, and Park, 1996; Marett and Wright, 1996) and simulated annealing versus genetic algorithms (Kim and Kim, 1996). Some three way comparisons are also found for the problem of balancing hydraulic turbine runners (Sinclair, 1993), project scheduling (Lee and Kim, 1996) and flow shop scheduling (Murata, Ishibuchi, and Tanaka, 1996). Unfortunately, conclusions in these studies are based on simple observational comparisons of mean performance. None of the studies above reported the statistical significance of their conclusions or base their results on any statistical measure. Clearly, the need remains for a comparative study that incorporates rigorous statistical analysis.

1.2 Purpose

The aim of this dissertation is to provide an empirical comparison of these three heuristic algorithms using three variants of the facilities location problem: capacitated (CFLP), multiple periods (MP-FLP), and multiple commodities (MC-FLP). We selected the facilities location problem domain for several reasons:

(1) It is an important problem domain in the field of operations management;

(2) It provides challenging problems whose solutions can be represented in various ways; and

(3) The new heuristics have not been widely applied to these problems creating an opportunity for new contributions.

Specifically, we intend to address the following questions:

What is an appropriate problem representation that facilitates problem solution using the new heuristics?

For each heuristic, what is the relationship and effect of various levels of key parameters and the quality of solutions?

For a set of medium sized, benchmark problems, what is the quality of solutions for each heuristic?

For each heuristic, what is the relationship between time allowed for computation and quality of solutions?

For each heuristic, is there a relationship between the size of the problem and the computation time required to reach a good solution?

For randomly generated sets of large problems, is there a dominant heuristic for all forms of facilities location problems?

If not, is there a dominant heuristic for each form of the facilities location problem?

In Chapter 2, we present a tutorial explaining how each of the heuristics works. Along with the basic algorithm and associated terminology, we also illustrate their operation using a small example. In Chapter 3 we review the

pertinent literature. We first review papers that not only demonstrate the wide array of applications for these heuristics but also show a variety of techniques for implementation. We then review papers that have compared all three of the heuristics. In Chapter 4, we outline our research methodology. In Chapters 5, 6, and 7, we develop the necessary heuristics for each of CFLP, MP-FLP, and MC-FLP respectively. In Chapter 8 we conduct the actual comparative study, and in Chapter 9 we present our conclusions.

Chapter 2

HEURISTICS TUTORIAL

In this chapter we introduce the operation of each of the three general heuristic algorithms: tabu search, simulated annealing, and genetic algorithms. For each one, we outline their origin, their basic algorithm, and their associated terminology. To further illustrate how each heuristic operates, we walk through the early solution steps for a small sample problem. We take the sample problem from the uncapacitated facilities location problem (UFLP) domain. Since we use the same problem for each heuristic, we introduce it first.

2.1 A UFLP Sample Problem

A UFLP consists of a set of N potential locations from which a set of M customers must be supplied. There is a fixed cost for operating any given location, and a variable cost per unit to supply a customer from a given location. The objective is to select the subset of facilities that can serve all customers at the lowest total cost. Mathematically, this problem is formulated as a mixed integer program:

$$\min Z = \sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}x_{ij} + \sum_{i=1}^{N} F_i y_i \tag{1}$$

subject to

$$\sum_{i=1}^{N} x_{ij} = 1, \qquad j = 1,2,\ldots,M \tag{2}$$

$$y_i - x_{ij} \le 0, \qquad i = 1,2,\ldots,N \tag{3}$$

	Fixed Costs	1	2	3	4	5	6	7	8	9	10
A	$300	$2	$4	$5	$10	$6	$3	$8	$2	$8	$4
B	$210	$4	$5	$4	$11	$9	$11	$9	$12	$6	$11
C	$150	$12	$2	$3	$4	$10	$2	$7	$11	$ 10	$10
D	$200	$10	$8	$7	$2	$4	$12	$4	$5	$2	$2
E	$275	$6	$7	$12	$3	$6	$4	$5	$3	$4	$7
F	$325	$7	$10	$6	$5	$5	$6	$3	$2	$3	$5
Demand		10	30	10	40	20	10	40	20	10	30

Figure 1. UFLP Sample Problem

$$y_i \in \{0,1\} \qquad i = 1,2,\ldots,N \tag{4}$$

$$x_{ij} \geq 0 \qquad i = 1,2,\ldots N, \quad j = 1,2,\ldots,M \tag{5}$$

where

C_{ij} cost to satisfy customer j's demand from location i
F_i fixed cost of operating a warehouse at location i
x_{ij} fraction of demand for customer j shipped from location i
y_i $\{0,1\}$ indicator where $y_i=1$ implies a warehouse is located at location i

Figure 1 shows an instance of this problem in tableau form which we use in the subsequent tutorials. The problem consists of six potential locations and ten customers. The fixed costs, variable costs, and customer demands are as shown. Before the heuristics can solve this problem, a structure to represent a solution must be specified. An easy solution representation for the UFLP is a vector $\mathbf{s} = \{y_1, y_2, y_3, y_4, y_5, y_6\}$, where $y_i \in \{0,1\}$ represents whether the specific location i is in operation ($y_i = 1$) or not ($y_i = 0$).

2.2 Tabu Search

The initial concepts of a tabu search strategy were first introduced by Glover as early as 1977 in a paper that introduced the idea of oscillating assignment as a heuristic approach for solving integer programming problems (Glover, 1977).

Glover explains that up to this point two heuristic strategies prevailed. The first one was an outside-in strategy that started with a solution from outside of the feasible region and made moves that led directly into the feasible region and stayed there. A second strategy, inside-out, started with a solution within the feasible region and made improving moves until no moves were available except those that would lead outside the feasible region. Glover's oscillating assignment consists of applying these methods alternatively, allowing the inside-out strategy to move back outside the feasible region. In this way new areas of the solution space can be investigated. Glover formally presented the full tabu search strategy in a series of papers and tutorials that address the various aspects of the algorithm (Glover 1989; Glover 1990a; Glover 1990b; Glover 1993). These papers present the basic algorithm and discuss various technical elements such as neighborhoods and neighborhood sizes, tabu list types, tabu list size, and aspiration criteria.

Tabu search starts with a random, feasible solution to the problem. From this solution, a set of neighboring solutions is generated. A neighbor solution is generated through a pre-defined change to the incumbent solution. The change, also known as a *move*, is specified such that the resulting solution is feasible. A cost function is specified and used to evaluate the quality of each solution. From the current set of neighboring solutions, the best one is selected according to the cost function and becomes the new incumbent solution. A new set of neighboring solutions is generated from the new incumbent solution and the process repeats until some stopping condition is met.

Without any modifications, the process described above will find the nearest local optimum and stay there. The salient characteristic of tabu search is a flexible short-term memory of recent moves known as the *tabu list*. With a tabu list, the selection for a new incumbent solution becomes: select the best neighboring solution according to the cost function whose generating move is not on the tabu list. This strategy in essence prevents backtracking into local optima and at times forces acceptance of inferior solutions. The size of the tabu list is a critical parameter and determines the length of time moves remain unavailable. A list that is too short will result in cycling of solutions; a list that is too long will restrict too many moves and, therefore, the overall search.

As new incumbent solutions are selected, the strategy keeps track of the best solution found so far. If at any time a move on the tabu list would result in a solution that is better than the best one so far, the move's tabu status is ignored and the solution accepted immediately. This is formally known as the *aspiration criteria*. The complete algorithm is shown in Figure 2.

Formally, we can define a tabu search as a set of structures, functions, and parameters required to fully implement the associated algorithm:

$$TS = \{ \mathscr{R}, \mathscr{N}(), \mathscr{C}(), \mathscr{A}(), \mathscr{T}(), \mathscr{P} \}$$

Where

$\mathscr{R}$	is the solution representation structure
$\mathscr{N}()$	is the neighborhood structure;
$\mathscr{C}()$	is the cost function
$\mathscr{A}()$	is the aspiration criterion function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

```
Define move operation x that generates neighbor solutions from incumbent solution ;
Define cost function C(•) ;
Define tabu list length T_l = τ;
Generate initial solution s_o ;
Set "best so far" solution s_b = s_o ;
Set best cost C_b = C(s_o) ;
Set tabu list T = {∅};
Repeat
    Generate the set X = {x_1, x_2,...} of all possible moves from s_o ;
    Generate the set S = {s_1, s_2,...} of neighbor solutions from each move x_i ∈X ;
    Repeat
        Find best s_i ∈ S such that C(s_i) > C(s_j), s_j ∈S, j ≠ i (assumes maximizing objective);
        If corresponding x_i is not tabu (x_i ∉ T)
            s_o = s_i ;
            T = T + x_i ;
        Else
            If f(s_i) > C_b
                s_o = s_i ;
                T = T + x_i ;
            Else
                S = S - s_i ;
    Until s_o has been replaced;
    If |T| > T_l, remove oldest x_i from tabu list T;
    If f(s_o) > C_b
        s_b = s_o ;
        C_b = C(s_o) ;
Until some stopping condition
```

Figure 2. Tabu Search Algorithm

At first, $\mathscr{P} = \{\tau\}$, the length of the tabu list. If we use a maximum number of iterations as a terminating condition, then $\mathscr{P} = \{\tau, \omega\}$.

We now use the UFLP problem to illustrate the basic operation of the tabu search algorithm. As explained in the previous section, we use a vector to represent solutions. Our first task is to define the neighborhood of solutions and the associated moves. For a given UFLP solution, we can define a neighboring solution as one where the operating status of a single potential location has changed. Let $\mathbf{x} = \{i\}$ be the associated move indicating a change in status for location i. Thus, given incumbent solution $\mathbf{s} = \{1,1,0,0,0,1\}$, a move $\mathbf{x} = \{1\}$ results

in a neighboring solution $\mathbf{s}_1 = \{0,1,0,0,0,1\}$. The move indicated a change in status for the first potential location. Therefore, the neighboring solution differs from the incumbent solution only in the status of the first location. Note that for any incumbent solution in this example, there are six neighboring solutions. In general, for UFLP and the neighborhood structure we have defined, the size of the neighborhood is N, the number of potential locations.

The cost function is the one shown in (1) in the previous section. We use the traditional aspiration criteria: the tabu status of a move is ignored if the cost of the resulting solution is better than the cost of the best solution so far. The length of the tabu list will be set at two.

Our initial incumbent solution will be generated randomly by taking the first six random numbers from Table 1. The rule will be $y_i = 1 \text{ if } rnd_i > 0.5$, zero otherwise. Our starting incumbent solution $\mathbf{s}_0 = \{0,1,1,0,0,0\}$ has a cost $C(\mathbf{s}_0) = 1710$. We set our "best so far" solution and cost to these values also and set our tabu list $\mathbf{T} = \{\varnothing\}$. We will now iterate through the algorithm.

Iteration 1. By definition, we can make six moves to generate six new solutions from the incumbent solution. The set of possible moves is then $\mathbf{X}$=$\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$, where $\mathbf{x}_1 = \{1\}$ indicates a move changing the status of location one in the incumbent solution. By applying $\mathbf{X}$ to $\mathbf{s}_0$, we arrive at the set of neighboring solutions $\mathbf{S}$ = $\{\{1,1,1,0,0,0\}, \{0,0,1,0,0,0\}, \{0,1,0,0,0,0\}, \{0,1,1,1,0,0\}, \{0,1,1,0,1,0\}, \{0,1,1,0,0,1\}\}$. Of these solutions, $\mathbf{s}_4 = \{0,1,1,1,0,0\}$ has the "best" cost at \$1210. Since $\mathbf{x}_4 = \{4\}$ is not in the tabu list, we will let $\mathbf{s}_0 = \mathbf{s}_4$, be

Table 1. Random Numbers

0.172660	0.786573	0.449903	0.546213
0.889322	0.935427	0.523163	0.193679
0.834534	0.716905	0.796399	0.162223
0.009002	0.025920	0.080756	0.482895
0.303151	0.634895	0.262880	0.015289
0.053135	0.687167	0.225513	0.960717
0.046045	0.222020	0.196931	0.763701
0.878358	0.494333	0.816749	0.527953
0.560228	0.246860	0.104934	0.306651
0.747382	0.979987	0.621410	0.884274
0.243208	0.644853	0.036288	0.992566
0.598157	0.038312	0.899635	0.065116
0.217797	0.904357	0.168491	0.406552
0.520997	0.520354	0.832720	0.914466
0.393537	0.596678	0.528487	0.424163
0.182925	0.358830	0.285751	0.907070
0.427306	0.860678	0.610157	0.127950
0.739046	0.408201	0.911457	0.454823
0.150650	0.641046	0.857670	0.214733
0.982749	0.056209	0.864828	0.023002
0.058134	0.704880	0.161430	0.592212
0.054058	0.912767	0.147810	0.315352
0.550153	0.872777	0.250357	0.122942
0.420510	0.770607	0.742809	0.285305
0.503944	0.596445	0.398021	0.129142

our new incumbent solution and $\mathbf{T}=\{\{4\}\}$. Also, since our new incumbent is better than our "best so far," $\mathbf{s}_b = \mathbf{s}_0$ and $C_b = \$1210$.

Iteration 2. The set of possible moves remains the same as before. Elements in the tabu list are not considered at this point. By applying $\mathbf{X}$ to our new $\mathbf{s}_0$, we arrive at a new set of neighboring solutions $\mathbf{S} = \{\{1,1,1,1,0,0\}, \{0,0,1,1,0,0\}, \{0,1,0,1,0,0\}, \{0,1,1,0,0,0\}, \{0,1,1,1,1,0\}, \{0,1,1,1,0,1\}\}$. Of these solutions, $\mathbf{s}_2 = \{0,0,1,1,0,0\}$ has the "best" cost at \$1060. Since $\mathbf{x}_2 = \{2\}$ is not in the tabu list, we will let $\mathbf{s}_0 = \mathbf{s}_2$, be our new incumbent solution and $\mathbf{T}=\{\{2\},\{4\}\}$.

Also since our new incumbent is better than our "best so far," $\mathbf{s_b} = \mathbf{s_0}$ and $C_b =$ \$1060.

Iteration 3. By applying $\mathbf{X}$ to our new $\mathbf{s_0}$, we arrive at a new set of neighboring solutions $\mathbf{S} = \{\{1,0,1,1,0,0\}, \{0,1,1,1,0,0\}, \{0,0,0,1,0,0\}, \{0,0,1,0,0,0\}, \{0,0,1,1,1,0\}, \{0,0,1,1,0,1\}\}$. Of these solutions, $\mathbf{s_2} = \{0,1,1,1,0,0\}$ has the "best" cost at \$1210. However, $\mathbf{x_2}$ is in the tabu list and $C(\mathbf{s_2})$ does not meet the aspiration criteria. Therefore, $\mathbf{s_2}$ is discarded. The next "best" solution is $\mathbf{s_1}$ with a cost of \$1220. Since $\mathbf{x_1} = \{1\}$ is not in the tabu list, we will let $\mathbf{s_0} = \mathbf{s_1}$, be our new incumbent solution and $\mathbf{T} = \{\{1\},\{2\}\}$. Note that $\mathbf{x_4} = \{4\}$ has dropped off the tabu list since the maximum length of the list was set at two. Also since our new incumbent is not better than our "best so far," $\mathbf{s_b}$ and C_b remain unchanged..

Iteration three demonstrates the power of tabu search. An inferior solution was accepted in order not to backtrack to the incumbent solution found at iteration one. From this inferior solution, the solution space can be further investigated. The algorithm continues until a stopping condition, commonly a pre-set number of iterations. For this particular problem, the optimal solution is in fact the same one found at iteration two with a cost of \$1060.

The basic algorithm shown above can be enhanced with different, dynamic types of tabu lists. Techniques can be used that intensify the search around regions of the solution space that generate good solutions. There are also various techniques for implementing a long-term memory to diversify the search

into broad areas of the solution space. We will explore these techniques in more detail as we review applications of tabu search in the next chapter.

2.3 Simulated Annealing

The first concepts of simulated annealing were introduced in Metropolis, et al., (1953), as a Monte Carlo approach for estimating the thermodynamic equilibrium of solids at a given temperature. The idea is to start the solid with the state of its particles set at random. Particles are then selected at random, and their state changed. If the change reduces total energy, the change is accepted; but, if the change increases total energy, the changed is accepted using a probability function tuned to the desired temperature. This approach served as the basis for an algorithm that could be used to solve difficult combinatorial problems. Kirkpatrick, et al. (1983) and Černý (1985) independently developed the algorithm we now know as simulated annealing by applying the Metropolis algorithm to combinatorial problems. Unlike the Metropolis algorithm which is applied at a given temperature, these authors applied the algorithm in stages, starting at a high temperature and reducing the temperature control at each stage until a *frozen* state is reached. Kirkpatrick, et al., demonstrated the algorithm on the problem of locating circuits on a chip and also on the traveling salesman problem. Černý also demonstrated the algorithm on the traveling salesman problem. Since these early experiments, simulated annealing has received much attention and has been applied to a variety of combinatorial problems.

Like tabu search, simulated annealing begins with a randomly generated solution; but, unlike tabu search, only one randomly selected neighboring solution is considered at a time. The neighboring solution is generated in the same way, through a pre-defined change or perturbation to the incumbent solution, $\mathbf{s_0}$, resulting in a new, feasible solution, $\mathbf{s'}$. If the cost of the new solution is "better" than the cost of the incumbent solution, the incumbent solution is replaced. If the new solution is inferior in cost, it may still be accepted with some probability. The probability of accepting inferior solutions is based on some probability function. A commonly used function is:

$$\mathrm{P}(\delta) = \exp(-\delta / t) \qquad (6)$$

where

δ is the cost difference between the two solutions, $C(\mathbf{s'})$- $C(\mathbf{s_0})$

t is a "temperature" parameter

The rate at which inferior solutions are accepted is controlled by the temperature parameter, t. This parameter is initially set to a high value to allow free exploration of the solution space and gradually decreased over time. The rate at which t is decreased is called the *cooling schedule*, and the time between each update of t is called an *epoch*. The cooling schedule could be an exponential or geometric decay function such as $t = \alpha t$, where $\alpha < 1$ or a linear decay function such as $t = t - c$, where c is a constant amount. Other functions are of course possible. During each epoch, one or more iterations of the perturbation process may be accomplished. The algorithm terminates whenever a stopping condition is met. The full algorithm is summarized in Figure 3.

```
Define a neighborhood function N(s0);
Select probability function P(δ) = exp(-δ/t);
Select initial temperature t_o;
Select a cooling schedule α(t);
Generate initial solution s_o;
Repeat
    Repeat
        Randomly select s ∈ N(s0);
        δ = f(s) - f(s_o);
        If δ < 0
            s_o = s;
        else
            x = random(0,1)
            if x < exp(-δ/t) then s_o = s;
    Until η iterations
    Set t = α(t);
Until stopping condition;
```

Figure 3. Simulated Annealing Algorithm

As we did for tabu search, we can formally define a simulated annealing algorithm as follows:

$$SA = \{ \mathscr{R}, \mathscr{N}(), \mathscr{C}(), \mathscr{S}(), \mathscr{T}(), \mathscr{P} \}$$

where

$\mathscr{R}$	is the solution representation structure
$\mathscr{N}()$	is the neighborhood structure;
$\mathscr{C}()$	is the cost function
$\mathscr{S}()$	is the cooling schedule
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

At first $\mathscr{P} = \{t_0, \eta\}$, the initial temperature and number of iterations per epoch. Other control parameters may be required for the cooling schedule and terminating condition functions.

We now use the same UFLP shown in Figure 1 to illustrate how simulated annealing works. We use the same vector representation for our solutions as before and we use (6) as our probability function.. For our cooling schedule, we use a simple geometric schedule, $t_{i+1} = \alpha t_i$, with α = 0.95. We set our initial

temperature t_0 at 145 and the number of iterations per epoch, η, to two. We address how these parameters are selected in the next chapter.

The initial solution is generated in the same way as in the tabu search example, randomly setting the status of each potential location. Since the same random numbers would be used (the list is reset for each algorithm), the initial solution will be the same, $\mathbf{s}_0 = \{0,1,1,0,0,0\}$ with a cost $C(\mathbf{s}_0) = 1710$.

Epoch 1, Iteration 1. A neighboring solution $\mathbf{s}'$ is randomly selected by changing the status of one of the potential locations. The element to change is randomly selected using the function $x = \text{ceil}(n * \text{rnd}(\bullet))$, where the ceiling function returns the next largest integer, *n* is the number of elements in the solution, and the random number function returns a uniformly distributed number in the range (0,1). Since our next random number is 0.046045, x = 1, $\mathbf{s}'$ = $\{1,1,1,0,0,0\}$, and $C(\mathbf{s}')$ = \$1570. Since this is a better solution, we accept it as the new incumbent.

Epoch 1, Iteration 2. Our next random number is 0.878358. This results in x = 6, $\mathbf{s}'$= $\{1,1,1,0,0,1\}$, and $C(\mathbf{s}')$ = \$1685. Since this is an inferior solution, we must consider whether to accept it based on our probability function. We will accept this solution if the next random number is less than P(δ) = .452438. Since our next random number is 0.560228, the incumbent solution remains unchanged.

Epoch 2, Iteration 1. At this point we decrease the current temperature t_0 = 145, to 0.95t_0 = 137.75. With our next random number, 0.747382, x = 5, $\mathbf{s}'$= $\{1,1,1,0,1,0\}$, and $C(\mathbf{s}')$ = \$1705. Again, since this is an inferior solution, we must

test for acceptance. With P(δ) = 0.375297 and our next random number 0.243208, we accept this inferior solution as our new incumbent.

The simulated annealing algorithm continues in this fashion, keeping track of the best solution found overall. The algorithm would stop when some condition is met, such as the temperature falling below some parameter $T_{\min}$. We will discuss further enhancements in the next chapter.

2.4 Genetic Algorithms

Holland (1975) formally introduced genetic algorithms as a general theory of adaptive algorithms based on the biology of reproductive systems. Holland's reproductive plan starts with an initial population that evolves over time. In the language of genetic algorithms, a population is a set of incumbent solutions, solutions are called chromosomes, and the elements or variables in each solution are called genes. Four genetic operators control evolution: selection, crossover, inversion, and mutation. These operators are applied probabilistically to each element of the population. Considering a vector usually represents a solution or chromosome in genetic algorithms, we explain the genetic operators this way:

Selection: a survival-of-the-fittest operation that assigns a probability of survival from one generation to the next to each chromosome in a population. The assigned probability is based on the fitness of the chromosome relative to the other chromosomes in the population. In practical terms, fitness is a mapping function that translates a solutions' objective function value to some probability in (0,1), or $\mathscr{F}: C(\bullet) \rightarrow (0,1)$.

Crossover: a recombination operation that generates a new solution or chromosome from portions of two other incumbent solutions (parents). The new solution (offspring) will consist of the head portion from the first parent, and the tail portion from the second parent. The point at which the parents are split to create head and tail portions is called the crossover point and is selected randomly. The first parent is normally selected at random, with chromosomes having a probability of selection in proportion to their fitness values. The second parent is also selected at random, but with all chromosomes having an equal probability of selection.

Inversion: a reordering operation that reverses the order of elements in the selected incumbent solution or chromosome.

Mutation: an operation that randomly alters genes in chromosomes. In its simplest form, a mutation may simply change the value of a single variable in an incumbent solution. Many other mutations are possible such as swapping the value of two variables in a solution, or reordering the values of a few variables in a solution.

There are two ways in which generation of solutions evolve. One way is to maintain a single population of solutions. As genetic operators are applied to the members of the population, the resulting offspring replace other members in the population, thereby maintaining the size of the population fixed. The member to be replaced may be selected at random from the existing population, or may be selected using an elitist approach where the member with the lowest fitness value is replaced. An alternative way to evolution is to create a new population

for each generation. Using this approach, offspring resulting from genetic operators are copied to the new population. When the new population has reached its proper size, the old population is discarded and the new population becomes the incumbent generation. In general, genetic algorithms continue this evolution process until they converge to a single chromosome populating the whole generation, or until some fixed number of generations have elapsed. The full algorithm is shown in Figure 4.

As we did before with tabu search and simulated annealing, we now define genetic algorithms as:

$$GA = \{ \mathscr{R}, \mathscr{F}(), \mathscr{X}(), \mathscr{M}(), \mathscr{T}(), \mathscr{P} \}$$

Where

$\mathscr{R}$	is a solution representation
$\mathscr{F}()$	is the fitness function
$\mathscr{X}()$	is the crossover operation function
$\mathscr{M}()$	is the mutation operation function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

At first $\mathscr{P} = \{\psi, \pi_x, \pi_m, \omega\}$ where ψ is the population size, π_x is the probability of applying the crossover operator, π_m is the probability of applying the mutation operator, and ω is the number of generations before termination. Other parameters may be required for a specific implementation.

We now use the UFLP in Figure 2 to illustrate how genetic algorithms work. We use the same vector representation and cost functions as before. Our fitness function assigns a selection probability to each chromosome in a generation such that chromosomes with lower cost will have a proportionally higher probability:

```
Define a cost function C(•);
Define a fit function F(•);
Define crossover operation ⊗;
Define a mutation function M(•);
Select a population size ψ;
Select a crossover rate π_x;
Select a mutation rate, π_m;
Generate a staring population S;
Repeat
  S_new = {∅};
  Repeat
      Randomly select p_1 ∈ S, based on F(C(s)), s ∈ S;
      Randomly select p_2 ∈ S, p_2 ≠ p_1, based on F(C(s)), s ∈ S;
      If Random(•) ≤ π_x s = p_1 ⊗ p_2 else s = p_1;
      If Random(•) ≤ π_m then s = M(s);
      If C(s) best so far, s_best = s;
      S_new = S_new + {s};
  Until |S_new| = ψ;
  S = S_new;
Until max_generations or other stopping condition;
```

Figure 4. Genetic Algorithm

$$F(\bullet) = \frac{\max\{C(s)|s \in S\} - C(\bullet)}{\sum_{s \in S} \max\{C(s)|s \in S\} - C(s)} \tag{7}$$

Note that the worst solution will receive a selection probability of zero. Our crossover operation consists of randomly selecting a point where each parent vector will be split. The offspring consists of the first portion of the first parent and the second portion of the other parent. For simplicity, we use a population size of five and no mutations are allowed. The crossover operator is applied with 100% probability.

We first generate a starting population of five solutions. Each solution is generated as shown in §2.2, using the random numbers in Table 1. Our starting

Table 2. Genetic Algorithm Starting Generation

s_i	$C(s)$	$F(s)$
{0, 1, 1, 0, 0, 0}	$1710.00	.0062
{0, 1, 1, 1, 0, 1}	$1435.00	.3522
{0, 1, 0, 0, 0, 1}	$1415.00	.3773
{0, 1, 0, 0, 1, 0}	$1505.00	.2641
{1, 1, 1, 1, 0, 1}	$1715.00	0

generation is shown in Table 2 including the corresponding cost and fit values. To create our second generation, we must create five new offspring solutions from this starting generation.

To create the first child solution we must select two parents based on their fit values. Our next available random number is 0.687167, since we used the first 30 numbers to create our starting generation. We select as our first parent, $\mathbf{p}_1$, the nth solution, such that n is the smallest number for which the sum of fit values for the first n solutions exceed our random number. Using this method, our first parent is $\mathbf{s}_3$. Our second parent, $\mathbf{p}_2$, is selected similarly, using the next random number 0.222020. This results in $\mathbf{s}_2$. To accomplish the crossover operation, we use the next random number, 0.494333, with the function $x = \text{ceil}(n * \text{rnd}(\bullet))$ to select a point between one and six (the point where the solution vectors will be split). Since this function evaluates to three, our child solution will consist of the first two elements of $\mathbf{p}_1$, and the last four elements of $\mathbf{p}_2$. Our child solution is $\{0, 1, 1, 1, 0, 1\}$, which is the same as $\mathbf{s}_2$ in our current generation, indicating that $\mathbf{s}_2$ will survive into the new generation. The next four children of the new generation

Table 3. Genetic Algorithm Second Generation

s_i	C(s)	F(s)
{0, 1, 1, 1, 0, 1}	$1435.00	0.4000
{0, 1, 1, 0, 1, 0}	$1535.00	0.0000
{0, 1, 1, 0, 1, 0}	$1535.00	0.0000
{0, 1, 0, 0, 1, 0}	$1505.00	0.1200
{0, 1, 0, 0, 0, 1}	$1415.00	0.4800

are created similarly and the whole new generation, along with cost and fit values, is shown in Table 3.

The genetic algorithm continues by creating one generation after another until a stopping condition is met. Looking at Table 3, two observations should be made about this example. First, the three best solutions from the starting generation survived to the new generation. Second, one of the new solutions has been duplicated, reducing the diversity in the pool of parent chromosomes. Fortunately, by our fitness function, these solutions will not be selected since they are the worst of the generation. Unfortunately, the remaining solutions have a *schemata* or pattern of the form {0, 1, *, *, *, *}, and the optimal solution is {0, 0, 1, 1, 0, 0}. Clearly, without mutation, the optimal solution cannot be reached from the current pool of parent chromosomes. The first observation shows the strength of genetic algorithms (survival of the fittest solutions) and the second observation a potential weakness that must be considered. In the next chapter we will review the pertinent literature that addresses these and other issues, including variations on the algorithm.

2.5 Chapter Summary

In this chapter, we first developed an example of an uncapacitated facilities location problem (UFLP) that we used to illustrate the operation of each general heuristic algorithm. For each heuristic, we then outlined a short history of its development and introduced the basic strategy. We then walked through the initial steps of each heuristic using the UFLP example.

The next chapter will consist of a literature review covering publications on the theory and application of each of these heuristics. We are interested in finding out the extent to which these heuristics have been used to solve business problems and the various techniques that have been used and developed in implementing these heuristics. We will also review in detail several comparative papers so we can establish the state of the art in the use of these heuristics.

Chapter 3

LITERATURE REVIEW

The objective of this chapter is to review the pertinent literature on tabu search, simulated annealing, and genetic algorithms. For each heuristic strategy, we will review papers to see how the methods have been applied to solve various problems with a specific interest in problem representations, technical implementation, and computational experience. We will conclude with a review of the few papers that report on comparative studies between all three of these heuristics.

3.1 Tabu Search

As we noted in the previous chapter, Glover introduced the full tabu search strategy in a series of papers and tutorials that address the various aspects of the algorithm (Glover 1989; Glover 1990a; Glover 1990b; Glover 1993). In this section, we review some of the published papers that apply tabu search to various combinatorial problems. A summary of these papers is shown in Table 4.

3.1.1 Assignment Problems

The classic quadratic assignment problem (QAP) is one of many hard combinatorial problems where tabu search has been used successfully. Taillard (1991), presents an efficient tabu search algorithm for this problem along with a

Table 4. Tabu Search Articles Reviewed

Author(s)	Year	Domain
Taillard	'91	QAP
Laguna, et al.	'95	Assignment
Gendreau, et al.	'94	VRP
Renaud, et al.	'96	VRP
Hindi	'95	Lot Sizing
Barnes and Chambers	'95	JS Scheduling
Barnes and Laguna	'93	JS Scheduling
Icmeli and Erenguc	'94	Project Scheduling
Rolland, et al.	'96	p-Median
Skorin-Kapov & Skorin Kapov	'94	Location/Allocation
Crainic, et al.	'93	Location/Allocation

comparison of previous results from earlier implementations of tabu search, simulated annealing, and other heuristics. Taillard's implementation of tabu search incorporates a random sized tabu list. A typical decision with tabu search is trying to determine a tabu list size that will prevent cycling without being too restrictive. Taillard's approach is to establish a range for the size of the list between $s_{\min}$ and $s_{\max} = s_{\min} + \Delta$ where Δ is some constant, and then change the size of the list every $2 * s_{\max}$ iterations to a random size within the range. Taillard found that using the random tabu list size resulted in a more reliable algorithm that found the best solution in about 30% fewer iterations than with a fixed size list. His numerical results show near-optimal solutions for problems of size up to $N = 100$ that are better than those found by any other algorithm, including earlier implementations of tabu search.

Laguna, et al. (1995) used tabu search to find solutions for the multilevel generalized assignment problem (MGAP). This problem consists of assigning n tasks to m agents at each of i levels of efficiency. At each level, each task n must be assigned to at most one agent, and each agent may have more than one task

assigned. However, each agent has a limited amount of a single resource, and each task requires some amount of that resource dependent on the agent and the level of efficiency. A network representation is used where tasks and agents are nodes and arcs between nodes represent assignments. In their implementation, the authors use a dynamic tabu list where an element remains tabu for a time based on its own attributes rather than on a fixed sized list. Specifically, when an arc is removed from the network and another inserted, the leaving arc remains tabu for a time. The tabu time is based on the number of alternate arcs in the same node, the difference in ranks between the leaving and the entering arcs, and the frequency with which the leaving arc has been selected previously. To benchmark their algorithm, a set of 100 single level test problems with 5 agents and 20 tasks were solved to optimality with a branch-and-bound algorithm. Tabu search was able to find the optimal solution in 94 of these cases. Using slightly larger single level problems with 5 agents and 25 tasks that could not be solved to optimality, Laguna, et al. showed that their tabu search algorithm found better average solutions than the branch-and-bound algorithm for a given length of computation time. A similar experiment was conducted for multilevel problems. Allowing 180 CPU seconds for the tabu search and a MIP optimization package, the authors showed the tabu search algorithm found average solutions at least as good in 7 out of 9 cases. They noted that tabu search had greatest difficulty with sparse problems where the number of levels is larger than the number of agents.

3.1.2 Vehicle Routing Problems

Another class of problems where tabu search has been applied is the vehicle routing problem (VRP). Gendreau, et al., (1994), develop a tabu search algorithm for this problem that, unlike previous algorithms, allows infeasible solutions to become the incumbent solution as a way to diversify the search. To implement this technique, the authors added two penalty elements to the objective function, one for exceeding vehicle capacity and one for exceeding tour length. The penalties, however, are not constant throughout the search. Each penalty element is associated with a penalty factor: α for capacity and β for length. Every h iterations, if the previous h solutions were feasible with respect to capacity, then $\alpha = \alpha/2$; if they were infeasible then $\alpha = 2*\alpha$. The same operation is done for β with respect to tour length. The inclusion of these dynamic penalty factors results in a balance of feasible and infeasible solutions being accepted as the incumbent solution. Their computational experience on a set of benchmark problems, shows that tabu search dominates eight other classical heuristics and outperforms other simulated annealing and tabu search implementations (except for a tabu search algorithm due to Taillard that performs as well).

Renaud et al., (1996) present a variant of the vehicle routing problem which includes multiple depots. The objective of this problem is to find a set of routes where each route starts and ends at the same depot, a single vehicle visits each customer, vehicle capacity is not exceeded, and the travel duration of each route is limited. Renaud et al., report on a tabu search algorithm that uses three phases: fast-improvement, intensification, and diversification. The fast-

improvement phase is a basic tabu search that ends when no improvement has been found in the last θ iterations. The intensification phase looks for improvements through changes in the routes within each depot, and the diversification phase looks for improvements through changes in routes across depots. Computational results on 23 problems from the literature show the tabu search heuristic finds the best solution in 20 cases.

3.1.3 Lot Sizing

Hindi (1995) developed a tabu search heuristic for solving the single-item, capacitated dynamic lot-sizing problem. This problem is a variant of the classical lot-sizing problem, allowing for production runs to continue over several planning periods without a new set-up charge and allowing a reservation charge for keeping the facility on but idle (i.e., not available for producing other items). Hindi uses a basic tabu search algorithm, yet, computational results using two sets of problems from the literature showed tabu search found the optimal solution in each case. Furthermore, Hindi reports solution times were short and not related to structural aspects of the problem.

3.1.4 Job Shop/Flow Shop Scheduling

Barnes and Chambers (1995) present an algorithm to solve the job shop scheduling problem with an interesting approach for obtaining diversification. They begin by selecting a range for the size of the tabu list. In an initial phase, the tabu list size is set at the highest point in the range. The algorithm is executed as usual, except that every time the incumbent solution is improved overall, the solution is pushed down into a last-in, first-out stack. This phase of

the algorithm ends after a number of iterations are completed without an improvement in the solution. At this point the second phase begins. The top solution is popped off the stack and used as a starting solution for a new set of tabu searches each at a different level of tabu list size within the range defined. Each of these searches ends after a number of iterations without improvements in the solution, and each solution found with an overall improvement is pushed into the stack. This process continues until all solutions in the stack are searched at each level of tabu list size or until a preset time limit is reached. The reported test cases include problems with 10 machines and 10, 15, or 20 jobs, and problems with 15 machines and 15 or 20 jobs. Ranges for the short-term memory used were 11-13, 11-14, 12-14 and 12-15. Limits for CPU times were set at 225, 240, and 270 seconds. Their computational results show the tabu search heuristic generally dominates other heuristics reported in the literature.

Barnes and Laguna (1993) report on a tabu search algorithm for the multiple-machine, weighted flow time problem, where several jobs must be scheduled on several identical machines such that makespan is minimized. Their implementation of tabu search is quite routine. However, there are two kinds of moves allowed when generating neighbor solutions. A swap move changes the position of two jobs on two different machines, while an insertion move takes a job from one machine and inserts the job into another machine. At each iteration, the best move of either kind is made. In order to implement diversification, the authors consider the notion of *move distance*. That is, moves with longer distance explore broader regions of the solution space. With respect

to swap and insertion moves, the insertion moves are considered to be of longer distance. Thus, after a certain period of time, non-improving swap moves (of shorter distance) are disallowed in favor of insertion moves. A set of eight test problems with known optimal solutions from a branch and bound algorithm was used to benchmark the heuristic. Tabu search was able to find the optimal solution for all, and with one exception, the time required was an order of magnitude less than branch and bound. A second set of nine, more difficult problems showed the heuristic could find the same optimal solution as branch and bound when it finished, or as good a solution as branch and bound when it failed to finish within 60 hours. In addition, tabu search took two orders of magnitude less time to reach its best solution than branch and bound.

3.1.5 Project Scheduling

Icmeli and Erenguc (1994) develop a tabu search algorithm for solving the resource-constrained project scheduling problem. Initial solutions for their algorithm are generated through a separate heuristic that specifies the priority in which activities will be delayed whenever a resource constraint conflict occurs. A move in this context consists of scheduling an activity to finish one unit of time earlier or later. Thus, for N activities, there are at most $2N$ moves, some of which may lead to infeasible solutions. A penalty is added to the cost function for infeasible solutions based on the number of constraints violated and their degree represented by the value of the constraint's slack variable. Icmeli and Erenguc depart from traditional tabu search in their implementation of the tabu list, or S-List. Instead of declaring a move as tabu, the cost function value is saved as

tabu, thus preventing the selection of any other solution with the same cost as the incumbent. The authors call this version of the tabu search algorithm with only a short-term memory TABU-S. Computational experiments showed this version found the best solutions in the early stages, suggesting the need for a version with long term memory, TABU-L. The long-term memory is implemented by restarting the algorithm with different starting solutions. A second tabu list, the L-List, is kept with the values of these starting solutions, to prevent repeating a starting solution. In addition, the cost value for the best solution of each long term memory iteration is made a permanent member of the S-List. Both of these techniques avoid cycling into regions that have been previously explored. Experiments with a set of 50 problems showed that TABU-L performed as well or better than TABU-S in 41 of the problems, with TABU-S performing better in the other nine.

3.1.6 Location Problems

Rolland, et al. (1996) develop a tabu search algorithm for the uncapacitated p-Median problem. Their implementation is fairly straightforward. Solutions are represented by two disjoint sets: one for selected facilities ($\mathbf{S}$) and one for unselected facilities ($\{\mathbf{V}\text{-}\mathbf{S}\}$, where $\mathbf{V}$ is the set of all potential facilities). Moves consist of adding or dropping a facility to or from set $\mathbf{S}$. Facilities are placed on the tabu list only when added to the set of selected facilities. The time duration of a facility in tabu status is set randomly as in Taillard (1991). A form of long term memory is implemented using a frequency count of the number of times a facility has been added to set $\mathbf{S}$. Thus, to evaluate the cost of adding any facility

v, a penalty factor $\Pi(v) = k$*Freq(v) is added to the cost function. The constant k in this problem is set to the value of the largest distance between a source of demand and a facility. The authors also introduce Glover's concept of strategic oscillation. Following an add or a drop move, the solution may not be feasible since it may not consist of p facilities. Strategic oscillation is implemented by allowing a range around p for $|\mathbf{S}|$. Within this range, add or drop moves are selected randomly. On the boundaries, one or the other is forced to keep $|\mathbf{S}|$ within the range. Computational results on a set of 88 problems showed the tabu search algorithm finds the optimal solution in 66% of the problems with an average optimality gap of 0.5% and a maximum optimality gap of 6.0%. On a set of 12 larger problems, the tabu search algorithm found better solutions than two other heuristics in all but one case.

Skorin-Kapov and Skorin-Kapov (1994) present an interesting tabu search heuristic for solving the location of hub facilities problem. In this problem, given n interacting nodes in a network, p must be selected as hubs, and the remaining n-p facilities must be assigned to one of the hubs. The objective is to minimize the total cost of interaction between all facilities. A starting solution is generated by selecting the p facilities with the greatest flows, and assigning the remaining facilities to their closest hub. The heuristic actually consists of two tabu searches, one within another. The outer tabu search tries to find the best set of facilities to designate as hubs. A move consists of replacing one of the hubs with a non-hub facility. For a given solution, its neighborhood consists of $(n\text{-}p)p$ such moves. The tabu list consists of the facilities that have been recently replaced as

hubs. In order to select the best move, they must be evaluated. This evaluation is done with an inner tabu search heuristic. Given a fixed set of hubs from the outer loop, the inner heuristic searches for the best allocation of non-hub facilities to the fixed hubs. A move consists of moving a facility from its current hub to another. Thus, the neighborhood consists of $(n\text{-}p)(p\text{-}1)$ moves. The tabu list consists of the recent facilities whose moves were accepted. Both the inner and outer heuristics are limited to a maximum number of iterations. For the inner heuristic, maximum iterations were set between 25 and 33% of the neighborhood size, while for the outer loop, maximum iterations were set at 20% of the neighborhood size. To achieve diversification, the complete heuristic is executed for several, distinct starting solutions. A form of long-term memory is used to generate each starting solution. A frequency count is maintained of the number of times each facility has been selected as a hub. A starting solution consists of the p facilities with the greatest flow to frequency ratio. This ratio penalizes facilities that were previously selected as hubs often. Experiments with 3 invocations of the long-term memory showed no benefit since each invocation ended at the same solution. Computational experience showed the tabu search heuristic outperformed two other heuristics in solution quality and computation time.

Crainic, et al. (1993), studied a different version of the location/allocation problem with multiple commodities. The basis for this problem lies in the logistics problem of balancing the allocation of empty transportation containers of various types between customers and distribution depots. The containers are the

multiple commodities in this problem. The flow of containers is both ways, between customers and depots, and in between depots. The objective of the problem, however, is to find a set of depots to use for the redistribution of empty containers that will result in the least overall cost, where the cost includes a fixed cost of operating a depot and a variable cost based on flows and distances. A vector of 0-1 variables representing the status of each potential depot can then represent the problem solution. Two types of moves are defined: add/drop moves and swap moves. Add/drop moves simply imply changing the status of a depot to operational (add) or non-operational (drop), with a resulting change in the number of operating depots. Swap moves involve a simultaneous add and drop move, thus maintaining the same number of operating depots. The search strategy first applies add/drop moves until a local optimum is found that has not been improved over several iterations. At this point, swap moves are used to intensify the search with the current number of operating depots. This search ends when a local optimum is found that has not been improved over several iterations. The solution found at this point may be infeasible, in which case the algorithm returns to the add/drop moves phase. If the solution found after the swap phase is feasible, a new set of swap moves is executed, except that only improving moves will be accepted. When no more improvement moves are available, the algorithm returns to the add/drop moves phase. This overall algorithm, add/drop-normal swaps-strict swaps, is executed for a fixed number of iterations. Two tabu lists are maintained: one for the add/drop phase, and one for the swap phase. The add/drop phase places the depot corresponding to an

add/drop move on the list to prevent its status reversal too soon. The swap phase places the pair of depots swapped on the list, to prevent their status reversal. The tabu list for swap moves is emptied at the start of a swap moves sequence. Following the end of a swap moves sequence, the contents of its tabu list are transferred to the add/drop tabu list. This ensures a gradual change of the best solution found after the swap sequence. To achieve diversification, a long-term memory is implemented to restart the overall algorithm with different starting solutions. A frequency count is maintained of the number of times each depot changed status either way. At the restart point, the best solution found so far is modified to become a new starting solution. The modification consists of changing the status for the γ depots with the lowest frequency count. The parameter γ is based on problem size. Computational experience was based on a set of 20 test problems, 12 medium problems with 25 potential depots and 125 customers, and 8 large problems with 44 potential depots and 220 customers. In addition, for each of the two problem sizes, half had low fixed-to-variable cost ratios, and half had high fixed-to-variable cost ratios. Results of the tabu search heuristic were compared against a dual ascent heuristic. The dual ascent heuristic showed significantly better computational efficiency, but tabu search found a better solution in 12 cases and the same solution in another 2 cases. When a descent phase was added to the tabu search algorithm to ensure a local optimum from the best solution found, and additional four cases are improved over the dual ascent heuristic.

The articles we reviewed for tabu search demonstrate a wide range of applicability for this heuristic. More importantly, these articles report on a wide array of techniques for achieving good results with tabu search. Of these, implementation of dynamic length tabu lists and various implementations of long-term memory are of particular interest since they enhance the basic search strategy.

3.2 Simulated Annealing

Since the early experiments of Kirkpatrick et al. (1983) and Černý (1985) discussed in the last chapter, simulated annealing has received much attention and has been applied to a variety of combinatorial problems. Most applications use the same basic algorithm, with most variations focusing on the cooling schedule. We now review several applications to gain a better understanding of the state of the art in using simulated annealing. Table 5 summarizes a list of these articles.

On the theoretical front, Lundy and Mees (1986) show that simulated annealing can be formulated as a Markov process. They also prove that the algorithm can be made to converge arbitrarily close to the global minimum, albeit the time required may not be bounded by a polynomial function and may exceed that of an optimal algorithm. As an alternative, they recommend cooling schedule parameter settings the authors claim give good results in polynomial time. For an initial temperature t_0, the authors suggest $t_0 >> U$, where U is some upper bound on the difference in objective function values between any two feasible solutions. Such a t_0 implies nom-improving moves will have a probability

Table 5. Simulated Annealing Articles Reviewed

Author(s)	Year	Domain
Lundy and Mees	'86	Theoretical
Osman and Potts	'89	FS Scheduling
Ishibuchi, et al.	'95	FS Scheduling
Ben-Daya & Al Fawzan	'96	Machine Scheduling
Schmidt and Jackman	'95	Assembly Line
Chen & Srivastava	'94	Group Technology
Liu, et al.	'94	Location Allocation

of acceptance close to one when the algorithm starts. Subsequently, the temperature parameter should be updated with $t_{i+1} = t_i / (1 + \beta t_i)$, for some $\beta <<$ $1/U$. Since this schedule results in slow cooling, an epoch consisting of a single iteration is appropriate. Finally, for some acceptable solution error ε and an associated error probability α, the authors indicated the terminal temperature can be set at $t_f \leq \varepsilon / (\log(|\mathbf{X}|-1) - \log \alpha)$, where $\mathbf{X}$ represents the solution space. Using the above parameters, the algorithm will execute a number of iterations proportional to log $|\mathbf{X}|$.

3.2.1 Job Shop/Flow Shop Scheduling

Osman and Potts (1989) test four simulated annealing heuristics on the permutation flow shop problem. The problem consists of scheduling n jobs on m machines, where each job must be processed on all machines in order, each job has a different processing time on each machines, and the schedule selected will be used for all machines. The objective is to minimize makespan. The authors use the standard probability function $\exp(-\Delta / T)$ for their algorithm, and a cooling schedule due to Lundy and Mees (1986), where $T_{k+1} = T_k / (1 + \beta T_k)$. They propose a starting temperature $T_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{i,j} / (5mn)$ (the $p_{i,j}$'s represent

processing times), and a final temperature of T_K=1. The β parameter is set at $\beta = (T_1 - T_K) / ((K-1)T_1 T_K)$ and K is the proposed maximum number of iterations, which is determined based on problem size using a regression equation. The authors test two types of moves for generating a neighbor solution, interchange (I) and shift (S), and two orders in which to search the neighborhood, ordered (O) and random (R). The combinations of moves and search orders result in four simulated annealing heuristics: SA(I, O), SA(I, R), SA(S, O), and SA(S, R). An interchange move consists of selecting two jobs in the current solution and exchanging their positions. A shift move consists of selecting two jobs in the current solution, and placing the earlier one after the later one (a forward shift), or taking the later one and placing it before the earlier one (a backward shift). Which shift is performed depends on whether the first job selected comes before or after the second one. An ordered selection attempts moves in the same sequential order (i.e., (1,2), (1,3), ..., (2,3), etc.) and repeats the sequence when finished, whereas a random search picks each job pair in random order. Computational experience is based on 12 test problem configurations of all $m \in \{4, 7, 10, 20\}$ and $n \in \{20, 50, 100\}$ combinations. For each configuration, ten problem instances were generated with random, uniform processing times in (1,100). Results showed the SA(S, R) heuristic performed best. Compared against the NEH construction heuristic due to Nawaz, Enscore, and Ham (1983), SA(S, R) gives better quality solutions, but requires more computation time. When compared against two descent heuristics, SA(S, R) again provides better solutions but still requires more computation time.

Ishibuchi, Misaki and Tanaka (1995) propose a modification to the Osman and Potts (1989) heuristic reviewed above. Using the same SA(S, R) algorithm above, they propose generating K neighbors of the current solution per iteration instead of the traditional single neighbor for simulated annealing. With K neighbors available, the best one is selected and the typical acceptance rule is applied (accept if it improves the current solution, or with a probability if it is inferior). A second alternate method is proposed where the K neighbors are evaluated at random, and the first one that improves the solution is accepted. If none improves the solution, then the best one is accepted using the traditional acceptance criteria. The advantage of these modifications is that the algorithms become insensitive to the selection of a cooling schedule. They compare their algorithms to the Osman and Potts algorithm (equivalent to modified algorithms when $K = 1$), a multi-start descent algorithm, and to two tabu search algorithms that each implement the same modifications proposed. The comparisons are made such that each heuristic examines the same number of candidate solutions. A set of 600 test problems was used. Initial experiments showed that K should be set proportional to the problem size. The authors use $K = .5n$ (n = number of jobs). Comparisons with the best move modification show the multi-start algorithm is better for problems with 10 jobs, and simulated annealing is slightly better than tabu search for 20 and 50 jobs problems. Comparisons with the first move modification show the multi-start algorithm is better for 10 jobs problems, simulated annealing is better for 20 jobs problems, and the Osman and Potts algorithm is best for 50 jobs problems. Finally, results also show the

first move modification is better than the best move modification. It should be noted that the authors made comparisons based on average percent deviations from reference solutions without any apparent testing of significance based on the variability in solution quality.

Ben-Daya and Al-Fawzan (1996) report on a simple implementation of simulated annealing for the one-machine mean tardiness scheduling problem. The objective is to schedule n jobs on one machine to minimize mean tardiness. The authors define a perturbation as the switch in order of two randomly chosen jobs in the current solution. They use a probability function slightly different from others, $P_a = \exp(-\alpha * \Delta)$, where $\alpha = 10 + COUNT*STEP*2$, and $COUNT$ is an iteration counter and $STEP$ is set at two. The algorithm is terminated after 2000 iterations without an improvement. The authors use two sets of test problems. The first set consists of problems from the literature and is used for benchmarking. The algorithm finds the optimal solution for all of these problems. The second set is randomly generated. The algorithm outperforms two other heuristics from the literature on this set, and produces solutions within 0.75% of the optimal on the average with a maximum optimality gap of 1.49%.

3.2.2 Assembly Line Problems

Schmidt and Jackman (1995) present a multi-echelon implementation of simulated annealing to solve an assembly line problem. In their formulations, the assembly process consists of a set of tasks with precedent requirements. Each task also has a set of technical requirements for its completion. An assembly station can be one of several types, where each type consists of a set of

technical capabilities and an associated cost. The objective is to find the least cost configuration, including the number of assembly stations, the type assigned to each station, and the assignment of tasks to each station according to precedent and technical requirements. The authors developed a nested, three tier heuristic. The outermost tier sets the number of assembly stations, starting with the number of stations equal to the number of tasks and decreasing it by one in an iterative manner. The middle tier is executed for each iteration of the outermost tier and consists of a simulated annealing phase to select the best assignment of station types to the assembly stations. The innermost tier is executed for each proposed configuration of the middle tier, and consists of a second simulated annealing algorithm to assign tasks to stations. For a given number of assembly stations, the middle algorithm generates a starting solution by assigning to each station the type with the most capabilities. A move or perturbation consists of randomly selecting a station and randomly assigning it a different type. For each of these station configurations, the innermost tier generates an initial solution by assigning tasks to stations sequentially while adhering to the precedence and technical requirements. A solution perturbation in this tier consists of randomly picking a task and reassigning it to a different station while maintaining a feasible solution. The cooling schedules for both simulated algorithms have the same structure. Temperature changes are done after N_B solutions have been found or after N_A iterations ($N_B < N_A$), whichever comes first. Each algorithm is terminated if N_A iterations are reached for the last N_C temperature changes. Finally, the temperature is adjusted using an

exponential decay function, $T = T_0 F^{N_T}$, where F = 0.9 and N_T is the number of perturbations so far. The authors provide two small examples of how the overall heuristic works, but no computational experience is reported.

3.2.3 Group Technology

Chen and Srivastava (1994) develop two simulated annealing algorithms for the group technology problem. Given a set of m machines and n parts that must be processed on some of those machines, the problem can be described as a graph partitioning problem. Here, the machines must be grouped into some number K of disjunctive sets while maximizing some objective (the authors use a summed similarities objective in their formulation). For their first algorithm an initial solution is generated by randomly assigning each machine to one of K_{max} cells. Neighborhood solutions are generated by randomly selecting a machine and moving it from its present cell to a different, randomly selected cell. To establish a starting temperature, the authors use a preliminary binary search to find a T_0 that will result in an initial acceptance rate η_0, which is high enough so that most solutions are accepted during the early iterations of the algorithm. The binary search consists of running the simulated annealing program for t iterations at some temperature T. Let t^+ and t^- represent the number of these iterations with a positive increase and decrease respectively of the objective equation. Then the acceptance rate at the current value of T is computed as:

$$\eta = \frac{\exp(-\Delta Z^+ / T)t^+ + t^-}{t^+ + t^-}$$

where ΔZ^{+} is the average of the positive increases in the objective equation over the t^{+} iterations. If η is less than the desired acceptance ratio η_0, T is increased; otherwise, T is decreased. This search is repeated until η converges to η_0 within an allowable margin. During experimentation, these values were set at t = 5(K-1)m and η_0 = 0.9. The number of iterations at each temperature was set to the size of the neighborhood, m(K-1), and the temperature parameter was updated with $T_{l+1} = T_l / (1 + \gamma T_l)$ where $\gamma = \ln(1+\delta) / 3\,\mathrm{s}(T_l)$, δ controls the cooling speed, and s(T_l) is the standard deviation of the objective function values at the current temperature. The algorithm is stopped when a maximum number of iterations is reached, or if the objective value has not changed over some number of consecutive iterations. The simulated annealing process is followed by a greedy search to ensure a local optimum is found. The authors also develop an alternative two-stage simulated annealing algorithm, where the first stage consists of a separate, faster heuristic used to find an initial solution. From this solution, the annealing algorithm proceeds as normal, except the starting temperature is much lower. This starting temperature is found using a similar binary search as before, except the objective now is to find the temperature for which the initial solution is in equilibrium. Here, equilibrium means the expected value of changes in the objective equation over several iterations is zero (i.e., increases and decreases cancel each other out). Computational experience shows that both algorithms outperform a graph partitioning heuristic, and more importantly, the improvement increases with the size of the problem. As

expected, the simulated annealing algorithms are slower than the graph partitioning heuristic, and the first algorithm is slower than the second. Since both simulated annealing algorithms produce comparable solutions, the second one is preferred for its speed of execution.

3.2.4 Location Problems

Liu, Kao and Wang (1994) present a simulated annealing algorithm for the location-allocation problem with rectilinear distances. The objective is to find the best allocation of n customers to m new facilities that minimizes the weighted sum of distances traveled. Initial solutions are generated by randomly assigning customers to facilities. A neighbor is generated by the random reassignment of a customer to another facility. The initial temperature is set very high and reduced rapidly, T=0.5T after each temperature change, until a desired acceptance rate is achieved after which point the temperature is reduced gradually, T=0.85T. At each temperature level, the number of iterations is a factor of the neighborhood size (set to five during experimentation). The algorithm terminates when it is considered *frozen*. The algorithm is considered frozen when five consecutive temperature changes have elapsed where the acceptance rate at each failed to reach a threshold level set at 0.01 (i.e., less than 1% of neighbors were accepted at each temperature level). Computational experience on test problems from the literature against other heuristics shows the simulated annealing algorithm generated comparable solutions with less computation time.

The papers we reviewed above demonstrate the broad applicability of simulated annealing. More importantly, they demonstrate various techniques for

Table 6. Genetic Algorithm Articles Reviewed

Author(s)	Year	Domain
Leu, et al.	'94	Assembly Line
Bulgak, et al.	'95	Buffers in AAS
Leu, et al.	'96	Assembly Line
Chan and Tansri	'94	Facilities Layout
Houck, et al.	'96	Location/Allocation
Vignaux and Michaelwicz	'91	Transportation
Murata, et al.	'96	FS Scheduling

implementing an effective algorithm. Some of these techniques include various cooling schedules, initial temperature control settings, and terminating conditions.

3.3 Genetic Algorithms (GAs)

Since their introduction by Holland, work on genetic algorithms was limited to a small group of people and publications were limited to conference proceedings and journals on artificial intelligence. As a result, broad interest in genetic algorithms from operations research/management researches was slow to grow (Dowsland, 1996). Nonetheless, there is now a good body of research on the application of genetic algorithms to operations problems. Results of their effectiveness are mixed, but promising. We now review several papers that are interesting for showing the wide spectrum of applicability as well as their innovative approach to adapting the basic genetic algorithm methodology to their problem domain. The papers reviewed are listed in Table 6.

3.3.1 Assembly Line Problems

Leu, Matheson and Rees (1994) develop a genetic algorithm for the assembly line balancing problem with an objective equation that minimizes the number of workstations with maximum efficiency. Their solution representation consists of an ordered list of tasks that maintains the precedence requirements.

Given a required cycle time, the number of stations and efficiency can be determined from the ordered list by assigning tasks to workstation until the cycle time limit is reached for each workstations. The authors show that if all feasible permutations of such an ordered list are explored, the optimal assembly line configuration can be found. Their algorithm maintains a single population of chromosomes from which children are created. The starting population consists of a few random, feasible solutions. Given a list of allowable tasks (i.e., for which all precedents have been assigned), an initial solution is created by randomly selecting the next task to place on the ordered list of tasks. The list of allowable tasks is then updated and the next task randomly selected. This process continues until all tasks have been assigned. From the population of initial chromosomes, new children are created and added to the population until it grows to some predetermined size. At this point, new children replace their principal parent if they have a better fitness value, thereby maintaining a fixed sized population. The selection operation is done in proportion to the goodness of each chromosome's fit value as usual. The crossover operation is a two point crossover, where each point is selected at random. Given two parents, each is split into three sections: H1|M1|T1 and H2|M2|T2 (head | middle | tail). Replacing the middle section of each parent generates two children. A simple swap, however, is not possible since precedence requirements must be maintained to achieve to new feasible solutions. Accordingly, for the first child, the tasks contained in M1 are identified and replaced in the order in which they appear on the second parent. For the second child, the tasks contained in M2 are identified

and replaced in the order in which they appear on the first parent. Since the task ordering of each parent is feasible, the children are also feasible. At each iteration, a decision is made whether to do crossover or mutation. A mutation consists of selecting a single parent using the same selection criteria as above, and randomly selecting a mutation point. The mutation consists of reordering all tasks after the mutation point in the same random fashion as initial solutions are generated. The algorithms are stopped after 500 children have been created or after 100 consecutive attempts where the best fit value did not change by more than 1 percent. The authors conduct an extensive experiment where the initial population size, the final population size, and the mutation rate are varied at three levels each. In addition, they experiment with generating the initial population randomly as described above, or through the use of alternative heuristics from the literature. Their experiments show the use of heuristics and the mutation rate were significant. In particular, an initial population generated from 5 heuristics gave better results than using 3 heuristics or randomly generated solutions. Also, a higher mutation rate of 18% was better than 10 or 2%. The experiments are done with a set of 40 basic problems and five replications for each cell of their factorial design.

Bulgak, Diwan and Inozu (1995) report on the development of a GA for determining the buffer sizes of an asynchronous assembly system (AAS). As described, an AAS consists of several automated workstations arranged in a closed circuit with a fixed number of pallets in the system. The objective is to determine the optimal buffer size in front of each station that will maximize the

output rate. Since the problem is stochastic in nature based on the probabilistic incidence of machine stoppage and restart, the objective function value is estimated through a discrete event simulation that executes as a helper routine for the GA. The GA itself is fairly routine. The buffer size for each station is limited to a range between 1 and 15. Since the numbers in this range can be coded as a four position binary number, the solution itself is represented by a binary string of length 4*number_of_stations*. The crossover operator is applied at a rate of 60% and mutation at a rate of 3.3%. The population size is set at 30. The stopping criterion consists of limiting the number of generations to 15 as long as the average fit value and the maximum fit value differ by no more than 0.5%. If not, the algorithm continues. The authors report the results of various experiments; however, in the absence of benchmarking against known optimal solutions or other comparative studies, it is difficult to assess performance.

Leu, Matheson, and Rees (1996) develop a basic genetic algorithm to solve the problem of balancing mixed-model assembly lines. In mixed-model assembly lines, several models of a product are assembled simultaneously. Toyota assumes and it has been shown elsewhere that the best sequence of models on the line is cyclical; furthermore, the number of units of each model in a cycle is a multiple of the ratio of its demand to total demand. Thus, if a cycle consists of 2 model A's, 3 model B's and 1 model C, then a sequence might be: AABBBC. The objective of the mixed-model balancing problem is to find the permutation of this cycle that will result in the least overall variability in demand for subassembly and component parts, thereby also minimizing inventory required and balance

labor requirements. To solve this problem, the authors develop a genetic algorithm using the permutation vector of models for chromosomes. The fitness function assigns proportional selection probabilities based on an objective equation that measures variability in parts consumption. They use a one-point crossover operator which works this way: after the crossover point is determined, elements that appear in the tail of the first parent are removed from the second parent at random; the remaining elements of the second parent become the head of the offspring, and its tail is the tail of the first parent. Their mutation operator, swap head and tail, consists of picking a random mutation point, and swapping the head and tail of the chromosome. In their algorithm, recombination was applied 80% of the time, and mutation 20%. They experimented on a set of 80 problems with 5 and 10 model cases. The performance of the genetic algorithm was compared with Toyota's Goal Chasing Algorithm. On the average, the genetic algorithm found solutions that were 1.94% better.

3.3.2 Facilities Layout Problems

Chan and Tansri (1994) experiment with three different crossover operators while solving the facilities layout problem using genetic algorithms. The solution representation is a permutation list indicating which facility/machine goes into which position. The three crossover operators are taken from Oliver, Smith and Holland (1987) and are Partially Matched Crossover (PMX), Order Crossover (OX) and Cycle Crossover (CX). The first two operators are based on a two-point crossover where the genes within the crossover points are swapped between the two parents. For example, given two parents S1=2-4-5-|3-8-9|-6-1-7

and S2=3-9-8-|6-5-4|-2-7-1 where the crossover points are as indicated, a simple swap would result in S1'=2-4-5-|6-5-4|-6-1-7 and S2'=3-9-8-|3-8-9|-2-7-1. However, both of these solutions are not feasible due to the duplication of machines. The PMX and OX operators specify a strategy for resolving this duplication. The PMX operator maps the replaced genes (3, 8, 9) to the replacements, (6, 5, 4) and adjusts duplicated genes resulting in S1'=2-*9-8*-|6-5-4|-*3*-1-7. The reverse mapping is done for S2'=*6-4-5*-|3-8-9|-2-7-1. The OX operator uses a shifting operation to resolve the duplication. Starting with the gene after the second crossover point, any duplicated gene is cut, and the remaining ones shifted left. For S1, from the original sequence starting after the second cut, 6-1-7-2-4-5-3-8-9, the duplicated genes resulting from the crossover (6, 4, 5) are eliminated resulting in 1-7-2-3-8-9. This sequence is used to fill out the offspring starting after the second crossover point and rotating back to the start of the sequence for a feasible solution S1' = *3-8-9*-|6-4-5|-*1-7-2*. The operation is similarly done for S2'. The CX operator is entirely different and does not use crossover points. The offspring is created by ensuring the position of each machine is preserved from at least one of the parents. Thus, if the first position is determined from the first parent, 2-?-?-?-?-?-?-?-?, then the position for machine 3 will also have to come from the first parent since it occupies the first position on the second parent. Similarly, with 2-?-?-3-?-?-?-?-? the position for machine 6 must come from S1, resulting in 2-?-?-3-?-?-6-?-?. At this point, the placement of machine 6 dictates the placement of machine 2 should come from S1. However, machine 2 is already in the solution, and we have found a

cycle. Having found this cycle of positions on S1, we fill the remaining positions from S2 for a final offspring S1'=2-9-8-3-5-4-6-7-1. For a fitness function, the authors are concerned with obtaining an inverse correlation between the objective cost function and a measure of goodness over the range of possible objective function values. To achieve the relation, the authors use $y = (x^{-\log x})10^{12}$, where x is the cost of materials handling and y is the fitness value. Preliminary studies showed the CX operator converges early, is very sensitive to the size of the population, and does not perform well. The OX operator is the slowest to converge, is not sensitive to the size of the population, and works well with a large number of generations and a small population. The PMX operator is consistent and generally provides the best fit solutions. Since it also converges faster than OX, it is the preferred operator. Using the PMX operators, the authors develop and test GAs for three variations of the layout problems: with fixed machines, with excess space, and with fixed machines and excess space. Using a 10 machine problem and fixing 2, the fixed machine GA finds optimal layouts 20% of the time (2 out of 10 runs) with the worst solution deviating 10% from optimal while only searching 0.22% of the solution space. The excess space experiment uses a 25-position layout with 10 actual machines and 15 dummy machines (with zero load and cost factors). Using the optimal cost value for the fixed machine problem as a substitute benchmark, this GA provides solutions within 6% to 23% deviation while only searching (2.063E-19)% of the solution space. Using the same benchmark for the fixed machine, excess space experiment, the GA finds two solutions with costs less than the benchmark

while only searching (1.315E-12)% of the solution space. The authors conclude genetic algorithms are a reliable tool for solving this kind of problem.

3.3.3 Location Problems

Houck, Joines, and Kay (1996) report on a genetic algorithm for solving the location-allocation problem. In this paper, both the placement of new locations and the assignment of existing customers to those locations are decision variables. The objective is to place the new facilities such that an assignment of customers can be accomplished for a least weighted distance objective. Given a two-dimensional coordinate map, the placement of a new facility is represented by its real-valued coordinates (x, y). A chromosome consists of a string of n such coordinate pairs. The cost function is evaluated by a helper method, the alternate location-allocation heuristic. The fit value is assigned to each chromosome based on an ordered ranking of the cost function and on a normalized geometric distribution:

$$\mathrm{P}(i\text{th chromosome}) = q'(1-q)^{i-1}$$

where $q' = q/(1-(1-q)^N)$, q is the probability of selecting the best chromosome (set at 0.08), i is the rank of the chromosome, and N is the population size (80 in this case). The authors use three crossover operators: simple crossover which is the standard single point crossover; arithmetic crossover which produces linear combinations of the parents at complementary proportions; and a heuristic crossover which produces an offspring which is an extrapolation from the better parent along a vector joining the two parents. They also use four mutation operators: uniform mutation which randomly selects one of the x or y variables in

a parent and replaces it with a random, uniformly distributed value along the variable's upper and lower bounds; boundary mutation which selects a variable from the parent and sets it equal to its upper or lower bound; non-uniform mutation which is similar to the uniform mutation except using a non-uniform distribution; and multi-non-uniform mutation which applies the non-uniform mutation to all variables in the parent. The authors used test cases from the literature and compared the genetic algorithm to a random restart heuristic and a problem specific heuristic. Computational experience showed the GA outperformed the problem specific heuristic in all but the smallest cases. The GA also outperformed the random restart heuristic for the larger problems.

3.3.4 Transportation Problem

Vignaux and Michalewicz (1991) investigate new genetic operators for problems that are best represented as matrices. They develop two genetic algorithms for the transportation problem, one using a permutation vector to represent solutions, and another using a matrix. The transportation problem consists of m sources of supply with limited capacity, and n destinations each with a specified demand. The problem is represented by a m x n matrix, where each cell $c_{i,j}$ identifies the cost of shipping one unit from source i to destination j. A matrix also represents the solution, where each cell $x_{i,j}$ identifies the quantity to be shipped from source i to destination j. The objective is to find the allocation of demand to sources of supply with the least cost. The author's first algorithm, Genetic-1, uses a vector to represent the solution. The vector consists of a permutation of the numbers {1,2, ..., q}, $q=m*n$, where each number represents a

cell in the solution matrix. This representation is translated to a solution in the following manner: for each element c_k in the vector, determine the corresponding cell in the solution matrix, $i = [(c_k - 1)/n + 1]$, $j = [(c_k - 1) \bmod n + 1]$, and let $x_{i,j}$ = min (*supply*[*i*], *demand*[*j*]); adjust the supply and demand data and repeat until all demand has been allocated. Using this vector representation, Genetic-1 implements a PMX crossover operator as described previously, and a mutation operator that swaps the order of two elements in the vector. The authors also implement an inversion operator that reverses the order of a parent chromosome to generate an offspring. The second genetic algorithm, Genetic-2, uses a matrix to represent solutions. With this representation, there is no need for translation; however, new genetic operators must be developed. The new crossover operator takes two parent matrices and generates two intermediate matrices, DIV and REM, such that each element of DIV is half the sum of the two corresponding elements from the parent matrices (integer part only), and each element of REM is the remainder of this division. The resulting REM matrix will have an even number of 1's on each row and each column. From the REM matrix, two new matrices, REM_1 and REM_2 are generated such that REM = REM_1 + REM_2, and the sum of each row and column of REM_1 is equal to the sum of each corresponding row and column of REM_2. Two offspring are now generated by adding the DIV matrix to REM_1 and REM_2. An example is shown in Figure 5.

V_1

1	0	0	7	0
0	4	0	0	0
2	1	4	0	5
2	0	6	0	0

V_2

0	0	5	0	3
0	4	0	0	0
0	0	5	7	0
3	1	0	0	2

DIV

0	0	2	3	1
0	4	0	0	0
1	0	4	3	1
1	0	3	0	1

REM

1	0	1	1	1
0	0	0	0	0
0	1	1	1	1
1	1	0	0	0

REM_1

0	0	1	0	1
0	0	0	0	0
0	1	0	1	0
1	0	0	0	0

REM_2

1	0	0	1	0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0

V_3

0	0	3	3	2
0	4	0	0	0
1	1	4	4	2
2	0	3	0	4

V_4

1	0	2	4	1
0	4	0	0	0
1	0	5	3	3
1	1	3	0	1

Figure 5. Sample Crossover Operator for Matrices (Vignaux and Michalewicz, 1991)

The mutation operator works by rearranging selected elements of its parent. A sub-matrix **W** of size pxq is created, $2<p<m$, $2<q<n$. Then p rows and q columns are randomly selected from the parent and the cells at their intersections are used to fill the sub-matrix, maintaining their relative orders. The sum of each row and column in **W** now become the capacity and demand for this sub-problem. The cell contents are now rearranged using the translation algorithm from Genetic-1: all cells are selected in random order and the minimum of its remaining supply or available capacity is allocated. The rearranged cells are then put back into their corresponding places in the parent matrix. We use

the following example from the authors' article for illustration. Assume a parent matrix **V**:

0	0	5	0	3
0	4	0	0	0
0	0	5	7	0
3	1	0	0	2

Select at random rows 2 and 4, and columns 2, 3, and 5. Then the corresponding sub-matrix is:

4	0	0
1	0	2

The new capacities for rows 1 and 2 are now 4 and 3; and, the new demands for destinations 1, 2, and 3 are 5, 0, and 2. A new arrangement of allocations then could be:

2	0	2
3	0	0

Note that in this arrangement the sums of rows and columns remain the same. Replacing these cells back into the parent, a new offspring is created:

0	0	5	0	3
0	2	0	0	2
0	0	5	7	0
3	3	0	0	0

Vignaux and Michalewicz conducted experiments to compare Genetic-1 and Genetic-2. They used some randomly generated problems and others from the literature. All of them were small problems, the largest being 10 by 10. Their results showed Genetic-2 outperformed Genetic-1 in all cases suggesting a matrix representation may be superior to a permutation vector for this class of problems.

3.3.5 Flow Shop Scheduling Problem

Murata, Ishibuchi, and Tanaka (1996) investigate the performance of various crossover and mutation operators in genetic algorithms for the n-job flow shop scheduling problem. They represent the problem of scheduling n jobs on m machines as a permutation vector. Parent chromosomes are assigned selection probabilities using the following equation:

$$P(\mathbf{x}_i) = \frac{[f_M(\mathbf{Y}) - f(\mathbf{x}_i)]^2}{\sum_{x_j \in \mathbf{Y}} [f_M(\mathbf{Y}) - f(\mathbf{x}_j)]^2}$$

where $\mathbf{Y}$ is the set of potential parent chromosomes and $f_M(\mathbf{Y})$ is the value of the worst solution in $\mathbf{Y}$. The authors also experimented with an equation that did not square the differences, but they found it to be inferior. Ten crossover operators were explored of which six were found useful for the flow shop problem. The six crossover operators are:

(1) one-point crossover where the offspring inherits the elements left of the crossover point intact from the first parent and the remaining positions are filled in the order they appear in the second parent (Figure 6a);

(2) two-point crossover (version I) where the offspring inherits from the first parent the elements outside the crossover points and the inside elements are filled in the order they appear in the second parent (Figure 6b);

(3) two-point crossover (version II) where the offspring inherits from the first parent the elements inside the crossover points and the remaining elements are filled from the second parent (Figure 6c);

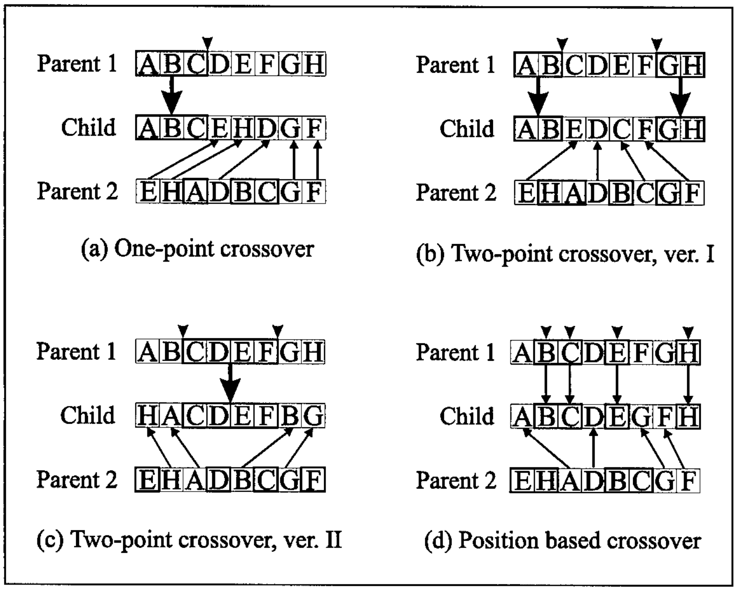

Figure 6. Crossover Operators (Murata, et al., 1996)

(4) two-point crossover (version III) which is a mixture of the two previous versions each applied with an equal probability;

(5) position based crossover (version I) where a randomly selected number of positions are inherited intact from one parent and the others filled in the order they appear in the other parent (Figure 6d); and

(6) position based crossover (version II) where each position is inherited from one parent intact with probability 0.5, and the remaining positions are filled as they appear in the other parent.

Four mutation operators are also investigated: (1) adjacent two-job change where the order of two sequential, randomly selected jobs is switched (Figure 7a); (2) arbitrary two-job change where the order of two randomly selected jobs is switched (Figure (7b); (3) arbitrary three-job change where the order of three randomly selected jobs is switched (Figure 7c); and (4) shift change, where a

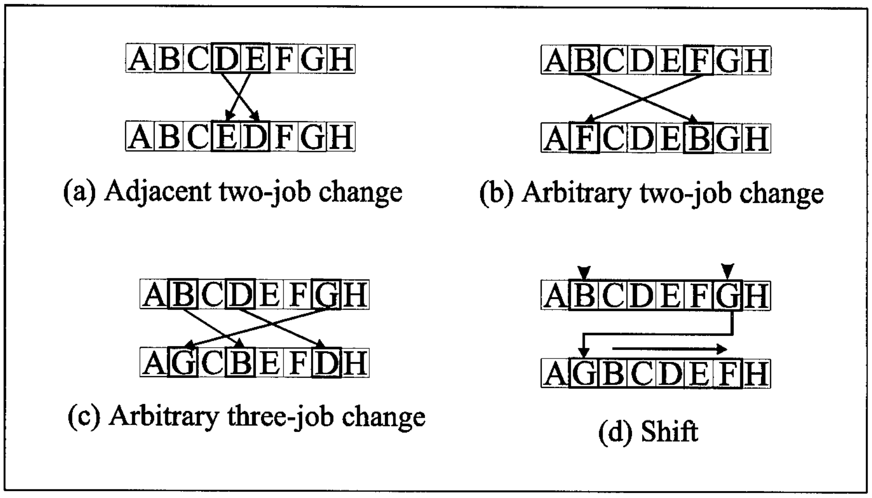

Figure 7. Mutation operators (Murata, et al., 1996)

randomly selected job is removed and reinserted at a different point in the vector causing a shift of the jobs between the two points (Figure 7d). The authors implement an *elitist* strategy whereby they remove a random chromosome from the new generation and replace it with the best chromosome from the parent generation. In their initial experiments, the authors found the two-point crossover (version I) and the shift change mutation operators performed best. It is also interesting that, unlike most other implementations where mutation is applied sparingly, a 100% mutation rate worked best! Using a set of 100 randomly generated problems with 20 jobs and 10 machines and a second set of 100 randomly generated problems with 50 jobs and 10 machines, the authors compared their genetic algorithm to a local search heuristic, a tabu search heuristic, and a simulated annealing heuristic. Their results indicate the genetic algorithm is somewhat inferior to the other algorithms.

As with tabu search and simulated annealing, genetic algorithms are broadly applicable. The most difficult barrier is finding an appropriate problem

Table 7. Comparative Studies Reviewed

Authors	Year	Domain	Rankings TS	SA	GA	Comparability Basis
Lee & Kim	'96	Project Mgt	2	1	3	CPU time
Sinclair	'93	QAP	1	2	3	Balanced time & solution quality
Kim & Kim	'96	Scheduling	-	1	2	CPU time
Kincaid	'92	Location	1	2	-	Best performance
Kuik, et al.	'93	Lot Sizing	2	1	-	CPU time
Marett & Wright	'96	Flow Shop	2	1	-	Information

representation along with effective genetic operators. For our research, insight into using permutation vectors and matrices are most useful and will be applied in the sequel.

3.4 Comparative Studies

In the papers reviewed so far, most authors conducted experiments to evaluate the performance of their algorithms. In most cases, benchmark experiments were conducted on small problems for which optimal solutions were known. Additional comparisons were made with results from other heuristic algorithms found in the literature, most of which are problem specific. In some cases comparisons were also made with one of the heuristic algorithms we are investigating in this research. However, none of the papers previously reviewed had as their principal objective an empirical evaluation or comparison of all three heuristic algorithms. In this section, we review several papers that do compare the performance of two or all three heuristics. Table 7 shows the articles reviewed.

3.4.1 Lee and Kim (1996)

Lee and Kim (1996) develop heuristic algorithms for the resource constrained project scheduling problem. The problem consists of scheduling n tasks, where

each task has precedence and resource constraints. The objective is to minimize total project completion time (makespan). Given a set of priorities for each task, the project can be scheduled using the following procedure:

(1) Determine the set of eligible tasks (i.e, activities whose precedence requirements have been met and for which enough resources are available).

(2) If all tasks have been scheduled, stop; else, if there are no eligible tasks, reset the current project time to the time of the first task completion, adjust the available resources data, and return to step (1); otherwise, move to step (3).

(3) From the set of eligible tasks, schedule the task with highest priority and adjust the available resources data; return to step (1)

This procedure is a priority scheduling procedure and will always result in a feasible schedule. All the procedure needs is a set of task priorities; different priorities will result in different schedules. The authors use this characteristic to represent problem solution. For all heuristics, a solution is represented by a vector of length equal to the number of tasks. Each element in the vector represents a task, and its value the priority assigned to the task. Initial solutions are generated by assigning random priorities to each task from a uniform distribution between 0 and 1. We now describe how each heuristic is implemented:

Simulated annealing: neighboring solutions are generated by switching the priorities of two tasks. The first task is selected randomly and the second task is selected within a range around the first task. The cooling schedule consists of an initial temperature that results in some percentage of uphill moves being

accepted (70%), an epoch length equal to a multiple of the number of tasks (20 times), and a geometric cooling rate (with parameter set at 0.99). The algorithm is stopped after 15 consecutive epochs where the acceptance rate was less than 0.5%.

Tabu search: neighboring solutions are generated in the same way as for simulated annealing. Since the neighborhood is fairly large, only a subset is investigated (equal to twice the number of tasks), and the best of those (subject to tabu restrictions) is selected as the new incumbent solution. The tabu list consists of the pairs of tasks involved in the most recent switches (the length set at 6). A long term memory is implemented which consists of maintaining a frequency count of the number of times each pair of tasks has been involved in a switch and using this count as a penalty factor in the objective equation (but only if the candidate solution is inferior to the incumbent solution). The algorithm is terminated after some number of consecutive iterations elapse without an improvement (set at 100).

Genetic algorithm: population size was determined so the computation time was approximately equal as in the other heuristics (set at 20). New generations are created by using a single point crossover operator and a mutation operator that switches the priorities of two tasks in each chromosome. Both operators are applied with certain probabilities (0.7 and 0.4 respectively). The algorithm is terminated after some number of generations elapse without improvement on the best solution (set at 30).

Computational experiments were conducted on two sets of problems, one set was randomly generated, and the other set comes from the literature and consists of 110 problems. The randomly generated sets consists of ten problems for each combination of number of activities (three levels: 50, 100, and 150), number of resource types (three levels: 1,4, and 7), and tightness of resource constraints (two levels: tight and not tight). This results in 180 test problems.

The number of activities factor did not affect the relative quality of solutions. In general, simulated annealing and tabu search had nearly the same performance, with the genetic algorithm lagging slightly behind, especially as the number of activities increased. However, the genetic algorithm had the least variability overall.

For the number and tightness of constraint factors, the results are the same as above. The genetic algorithm lagged slightly behind the other two heuristics, but had the least variability.

On the second set of problems from the literature, only the simulated annealing and tabu search heuristics were tested. After using the same parameters from the previous tests on this set, the parameters were updated for each heuristic. Both heuristics improved markedly after new parameters were employed, and simulated annealing performed better than tabu search.

Overall, simulated annealing and tabu search had similar results with the genetic algorithm slightly inferior. The authors point out that parameters were selected to equalize the amount of computing time rather than to get the best

results from the heuristics. They suggest that different results may be obtained if parameters were selected differently.

3.4.2 Sinclair (1993)

Sinclair compares the performance of five heuristic algorithms: the three we are researching plus Great Deluge Algorithms and Record-to-Record Travel. We will limit our discussion to the three in which we are interested. Sinclair develops heuristics for the hydraulic turbine runner balancing problem. In this problem, blades that may have slight variations in mass must be placed on the shaft of a turbine so as to minimize the unbalance. The problem is formulated as a quadratic assignment problem. The solution representation is thus a permutation vector that indicates the placement of each blade. This representation is used for all heuristics. Sinclair develops basic implementations of each heuristic and selects parameters that will result in the best performance in terms of solution quality and computation time. The following parameters are used:

Simulated annealing: the length of each epoch is determined by two parameters, a minimum number of candidate solutions that must be accepted (set at 30) and a maximum number of solutions that will be explored (set at 100). An epoch ends whenever either of these two parameters is exceeded.

Tabu search: the tabu list size is set at 15, 175 short-term memory iterations are allowed for each initial solution, and one initial solution is generated (one long-term memory pass)

Genetic algorithm: population size is set at 200. A modified one-point crossover operator is used and two-element swaps are used for a mutation operator. Crossover and mutation rates are set at 60% and 0.05% respectively.

Sinclair proposes the following methodology for the experimental evaluation of optimization algorithms:

(1) A problem class is selected as test domain for the algorithms.

(2) Data for different instances of the same problem class are collected.

(3) The algorithms are applied to the different examples in the problem class.

(4) Conclusions are drawn from the results.

Following the above methodology, Sinclair uses the hydraulic turbine runner balancing problem as a test domain. He uses a set of 37 real world problems for experimentation. Sinclair points out the following steps taken to avoid validity problems:

(1) The same data structures were used for all heuristics and the same procedure was used to construct initial solutions.

(2) Parameters were selected so that no heuristic had an advantage in any performance measure at the cost of another.

(3) The same person did all programming in the same environment.

(4) All experiments were run on the same machine.

The performance measures were objective function value and computation time. Algorithms that depend on random number streams were executed 100 times and their average used for comparison purposes. The first finding is that the genetic algorithm performed significantly worse than the other heuristics even

while requiring more computation time. Between simulated annealing and tabu search, tabu search provided the better solution in 28 out of 37 cases. Computation time for both of these heuristics was comparable.

3.4.3 Kim and Kim (1996)

This paper compares the performance of simulated annealing and genetic algorithms on a scheduling problem. The problem consists of scheduling the processing of products with a multiple-level structure consisting of finished goods, sub-assemblies, sub-sub-assemblies, etc. The objective is to schedule processing in order to minimize tardiness and earliness penalties, where tardiness is charged only to finished goods and earliness is charged to both finished goods and components. The penalties can be thought of as shortage and holding costs.

Their simulated annealing algorithm is controlled by three parameters: the cooling ratio, an epoch length parameter, and a terminating condition parameter. The cooling ratio determines how fast the temperature control parameter is reduced; three levels were explored: 0.75, 0.80, and 0.85. The epoch length is set to a maximum number of moves equal to a multiple of the neighborhood size; three factor levels were explored: 1, 3, and 5. The algorithm is terminated whenever five consecutive epochs terminate with the percentage of moves accepted less than the given parameter; four levels were explored: 0.1, 0.2, 0.3, and 0.4.

For the genetic algorithm, the authors fixed the population size to 80 and defined the terminating condition as 100 consecutive generations without

improvement.. They experimented with three levels for a mutation rate (0.001, 0.005, and 0.01), and four levels for a crossover rate (0.6, 0.7, 0.8, and 0.9). They also use a scaling factor in their fitness function.

After conducting parameter selection experiments, they selected five parameter combinations for each heuristic. Using a set of 150 randomly generated problems, they compared the performance of each heuristic. Their results show simulated annealing outperformed the genetic algorithm in all of the problems. They speculate the genetic algorithm did not do well because the problem representation and constraints were not well suited for taking advantage of the inheritance effects.

3.4.4 Kincaid (1992)

Kincaid considers the problem of locating p noxious facilities using two measures of distance: p-dispersion and p-defense-sum. The author develops conventional simulated annealing and tabu search heuristics for each of the two problems A second version of a tabu search heuristic is also developed where the initial solution is generated by a separate semi-greedy heuristic. They typical parameter setting experiments were conducted.

The author tested the heuristics on a set of 30 problems. In general, the author found the second version of tabu search performed best, followed by the conventional version of tabu search. Simulated annealing came in last even though it roughly took 2.5 to 3 times as long as tabu search.

3.4.5 Kuik et al. (1993)

In this paper, the authors present simulated annealing and tabu search heuristics for solving multilevel capacitated lot sizing problems. Given a multilevel product structure, the objective is to determine the setup times and run quantities that minimize setup and holding costs. The authors use a problem solution representation which consists of a 0-1 vector. Each element in the vector represents a possible setup in a given period for a particular product component. Given an incumbent solution, a neighboring solution is one where the setup status of a single element in the vector has changed while remaining feasible.

Unlike other studies which included a parameter setting experiment, the authors in this paper used their experience from previous studies to set parameters. Their simulated annealing was set to a maximum of 100 epochs with 50 moves allowed per epoch. They also used a conventional cooling schedule with a cooling rate of 0.95. Their tabu search algorithm was limited to 500 short-term memory iterations, a neighborhood size of 10, and a tabu list size of 10. The parameter settings ensured each algorithm had comparable computation times. They completed five replications of each heuristic on each of their 18 test problems.

In comparing the performance of the two heuristics, the authors conclude that simulated annealing did slightly better than tabu search. However, they caution that with different parameters, different results can be obtained.

3.4.6 Marett and Wright (1996)

Marett and Wright compare the performance of tabu search and simulated annealing for a multiple-machine, flow-shop scheduling problem. In this problem, several jobs must be scheduled on three machines. Once scheduled, all jobs follow the same sequence on each machine. Each job has its own processing time for each machine and a sequence dependent setup time. The problem is represented by a permutation vector and the neighborhood consists of all pair-wise swaps of elements in the vector.

The authors develop two general versions of the tabu search algorithm. One is the conventional best improvement algorithm which selects the best candidate solution from the set of neighboring solutions. An alternative version is a first improvement algorithm which while evaluating the neighborhood, selects the first candidate solution that is not tabu and improves the incumbent solution (if none improves the incumbent, then the best non-tabu candidate solution is selected as usual). The authors also consider four different tabu criteria. To select parameters, they conducted a Fibonacci search.

In making their comparisons, the authors allow each heuristic the same amount of information rather than the same amount of time. They test the heuristics on eight test problems. Among their findings, they authors conclude that the first improvement version of tabu search is better than the best improvement version. They also find that simulated annealing is sometimes better and sometimes worse than tabu search, but its superiority improves with the complexity of the problem.

3.5 Chapter Summary

In this chapter we highlighted several papers that are representative of the state of the art in applying tabu search, simulated annealing, and genetic algorithms to operations problems. Some of these problems include assignment problems, vehicle routing problems, job shop and flow shop scheduling, project scheduling, layout problems, and location/allocation problems. However, our principal interest in these papers was to survey the various techniques, enhancements, and variations used in applying each heuristic. These papers presented various ways to manage tabu lists, cooling schedules, problem representations, and genetic operators among others. This information is valuable in developing new algorithms to solve problems in new domains.

Chapter 4

RESEARCH METHODOLOGY

In this chapter, we outline the research methodology for an empirical comparison of three general heuristic algorithms: tabu search (TS), simulated annealing (SA), and genetic algorithms (GA). In the first section we re-iterate the research questions designed to achieve our principal research objective. In the following section we outline a four-stage study to answer these questions. Subsections describe the experiments and statistical tests used to answer each of the research questions.

4.1 Research Questions

Our basic research objective is to compare three general heuristic algorithms in search of a best performer. To conduct this study, facilities location was selected as a problem domain on which to test the heuristics. In particular, we study the performance of the three heuristics on three variations of facilities location problems: capacitated facilities location problems (CFLP), multiple periods facilities location problems (MP-FLP), and multiple commodities facilities location problems (MC-FLP). By using three problem variations we ensure a broader comparison over different problem structures and degrees of complexity. We selected facilities location for three reasons:

(1) It is an important problem domain in the field of operations management;

(2) Problems are difficult to solve optimally; and,

(3) To our knowledge, none of the heuristic algorithms we are studying have been previously applied to the three variants we use.

In this study we work with nine algorithms corresponding to the nine heuristic-problem type combinations. To achieve our research objective, the following research questions previously listed in Chapter 1 were developed:

1. What is an appropriate problem representation that facilitates problem solution using the new heuristics?

2. For each heuristic, what is the relationship and effect of various levels of key parameters and the quality of solutions?

3. For a set of medium sized, benchmark problems, what is the quality of solutions for each heuristic?

4. For each heuristic, what is the relationship between time allowed for computation and quality of solutions?

5. For each heuristic, is there a relationship between the size of the problem and the computation time required to reach a good solution?

6. For a new set of large problems, is there a dominant heuristic for each form of the facilities location problem?

7. For a new set of large problems, is there a dominant heuristic for all forms of facilities location problems?

4.2 Research Outline

To answer the research questions, we conduct a four-stage study. In the first stage we consider how best to represent solutions to each of the facilities location problems. In the second stage we conduct experiments to evaluate and select control parameters for each heuristic method. In the third stage we conduct benchmark experiments and experiments to evaluate the significance of

computation time and problem size. In the last stage, we conduct a statistical study to compare the heuristics against each other. We now describe each stage in greater detail.

4.2.1 Solution Representations

Tabu search, simulated annealing, and genetic algorithms are search heuristics. As such, they search the solution space by moving from one solution to another in some prescribed manner. Crucial to their operation is a representation of the problem's solution as it may affect the efficiency and effectiveness of the algorithm. A poor representation may result in slow searches of large solution spaces. Therefore, finding a good representation that facilitates the search process is important. Several representations are considered including a 0-1 vector, a permutation vector, and a matrix. We defer further explanation of these representations to chapters 5, 6, and 7 following a full description of each of the three facilities location problems. Having defined each representation, we test each of the nine heuristic-problem algorithms with each solution representation on a small problem set. For each of CFLP, MP-FLP, and MC-FLP, we select a solution representation that performs well across all three heuristics for further study. We consider a "good" solution representation one that strikes a balance between computation time and solution quality. The results from this stage will answer the first research question.

4.2.2 Parameter Setting Experiments

Once a solution representation is selected, each heuristic algorithm must be "tuned" by selecting appropriate control parameters. In general, these

parameters control the quality of solutions found by each algorithm. To select the proper parameters, full-factorial experiments are conducted for each heuristic on a single problem. For each of CFLP, MP-FLP, and MC-FLP, a single problem is selected for the experiments. The problem selected is of medium-size with average constraint values. The heuristic control parameters are tested at various levels following suggestions from the literature. Each factorial design is fully described in subsequent chapters.

Using the results from the experiments above, we use the Kruskal-Wallis H Test on each parameter tested for each heuristic to determine if various levels have a significant impact on solution quality. The results from these tests will answer the second research question. At the end of this set of experiments, we select parameter settings for each of the nine heuristic-problem combinations.

4.2.3 Benchmarking

The third stage of this research consists of benchmarking experiments. The objective is to measure the "optimality gap" for each heuristic algorithm on small and medium-sized problems for which optimal solutions can be computed. For each of the nine heuristic-problem combinations, the corresponding algorithm will be applied to a set of problems several times and the average optimality gaps reported. For these experiments, we use the set of parameters selected in the previous stage. Optimal solutions are obtained by adapting algorithms found in the open literature. Results from this stage will answer the third research question.

4.2.4 Time versus Quality

We use the results from the benchmarking experiments to find the relationship between time of computation and quality of solution. For each of the test problems used for benchmarking, we compute Spearman's rank correlation coefficient for each of the heuristics applied to the problem. We use the corresponding test of hypothesis to determine if a correlation in fact exists:

H_0: $\rho_s = 0$ (There is no population correlation between ranks)
H_a: $\rho_s \neq 0$ (There is a population correlation between ranks)

Results of these experiments will answer the fourth research question.

4.2.5 Size versus Time

Again, we use the results from the benchmarking experiments to determine if the size of a facilities location problem has an impact on the time each heuristic requires to find a good solution. For this experiment we use the Kruskal-Wallis H Test on each heuristic. For each heuristic, the statistics ranked are the average computation time required for each problem tested over several cases. The probability distributions we are measuring are those for various sizes of CFLP, MP-FLP, and MC-FLP. The following test of hypothesis will be evaluated for each heuristic:

H_0: The probability distributions are identical
H_a: At least two of the probability distributions differ in location

Results from these experiments will answer the fifth research question.

4.2.6 Global Comparison

The fourth and final stage involves a comparative evaluation of the three heuristics on larger size problems. The statistical study for this stage follows the

method of Dudewicz and Dalal (1975) described in Law and Kelton (1982). This method involves a two-stage sampling procedure to determine the total number of replications needed for a decision. The decision we want to make is which of k systems has the smallest response, and to make that decision with a probability of being correct of at least P^*. In our case, we want to select which of three systems (TS, SA, GA) results in the lowest average cost. The method also allows for a degree of indifference ($d^* > 0$) between mean responses. The greater the degree of indifference, the fewer number of replications needed to make a decision within the desired probability of correctness.

We apply the following procedure to each of CFLP, MP-FLP, and MC-FLP. In the first stage, we make $n_0 (n_0 \geq 2)$ replications for each heuristic and compute the first-stage sample mean and variance:

$$\overline{X}_i^{(1)}(n_0) = \frac{\sum_{j=1}^{n_0} X_{ij}}{n_0}, \text{ and } s_i^2(n_0) = \frac{\sum_{j=1}^{n_0}\left[X_{ij} - \overline{X}_i^{(1)}(n_0)\right]^2}{n_0 - 1}$$

for i = 1,2, and 3, representing TS, SA, and GA. Using this information, we can compute N_i, the total number of replications needed for each system (TS, SA, GA):

$$N_i = \max\left\{n_0 + 1, \left\lceil \frac{h_1^2 s_i^2(n_0)}{(d^*)^2} \right\rceil\right\}$$

where h_1 is obtained from a table (depending on k, P^* and n_0) and d^* is the degree of indifference we are willing to accept. After we make $N_i - n_0$ additional replications for each system, we obtain the second-stage sample mean:

$$\overline{X}_i^{(2)}(N_i - n_0) = \frac{\sum_{j=n_0+1}^{N_i} X_{ij}}{N_i - n_0}$$

and the weights

$$W_{i1} = \frac{n_0}{N_i}\left(1+\left\{1-\frac{N_i}{n_0}\left[1-\frac{(n_i-n_0)(d^*)^2}{h_1^2 s_i^2(n_0)}\right]\right\}^{1/2}\right)$$

and $W_{i2} = 1 - W_{i1}$, for i = 1, 2, and 3. Finally, we can obtain the weighted sample mean for each system,

$$\widetilde{X}(N_i) = W_{i1}\overline{X}_i^{(1)}(n_0) + W_{i2}\overline{X}_i^{(2)}(N_i - n_0)$$

and select the system with the lowest $\widetilde{X}(N_i)$.

The results from the procedure above in fact allow us to rank each heuristic for each test problem on which they are applied. Following the suggestions from Golden and Stewart (1985) for comparing the performance of heuristics, we use the Friedman F_r test for a randomized block design to test if any of the heuristics is in fact better than the others across all problems. We use the following test of hypothesis:

H_0: the probability distributions of the response rates for the heuristics are identical
H_a: at least two of the three distributions differ in location

If we reject the null hypothesis, we use the Wilcoxon signed rank test for paired difference designs to determine which distribution for the heuristics is in fact different. The results from these experiments will answer the sixth question when done within FLP problem types and the seventh and final research question when done across FLP problem types.

Since our intent is to make a selection among comparable systems, the answers for the sixth and sevenths research questions will in fact be made with respect to the computation time and information allowed each heuristic. Specifically, we select the best heuristic when all are allowed comparable computation time, and when all process a comparable amount of information.

4.3 Chapter Summary

In this chapter we outlined our research methodology for comparing the performance of three heuristic algorithms: tabu search, simulated annealing, and genetic algorithms. Six research questions were developed and a four-stage study to answer these questions was outlined. In the first stage we explore problem solution representations. In the second stage we select parameters for each heuristic that make them comparable. In the third stage we conduct benchmarking experiments. In the final stage, we compare the heuristics performance against each other. Due to the nature of the data, we use non-parametric statistical tests to obtain our results.

Chapter 5

HEURISTICS FOR CAPACITATED FACILITIES LOCATION PROBLEMS

In this chapter we present the development of tabu search, simulated annealing, and genetic algorithm heuristics for solving the capacitated facilities location problem (CFLP). In the first section we introduce the CFLP, present a formulation, and mention some of the relevant literature. In the following section we present the three solution representations we consider for CFLP. We then present the heuristic algorithms. For each one, we specify our particular implementation and parameters for each representation. After all three heuristics have been explained, we report the results from the first three stages of experimentation: selection of appropriate representation, parameter selection and effects, and benchmarking.

5.1 The Capacitated Facilities Location Problem

For CFLP, we wish to determine which of N capacity constrained locations to operate so we can satisfy the demand for M customers at the lowest sum of fixed and variable costs. Finding optimal solutions for this problem usually involves an implicit enumeration algorithm such as branch and bound. Exact approaches for this problem can be found in Davis and Ray (1969), Ellwein and Gray (1971) and Akinc and Khumawala (1977). Due to the difficulty of solving this problem, domain specific heuristic methods have also been developed such as Sa (1969)

and Khumawala (1974). The formulation of this problem is similar to the formulation for the uncapacitated facility location problem we saw in Chapter 2 with the additional constraint for capacity:

$$\min Z = \sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}x_{ij} + \sum_{i=1}^{N} F_i y_i \tag{8}$$

subject to

$$\sum_{i=1}^{N} x_{ij} = D_j, \qquad j = 1,2,\ldots,M \tag{9}$$

$$\sum_{j=1}^{M} x_{ij} \le S_i y_i \qquad i = 1,2,\ldots,N \tag{10}$$

$$y_i \in \{0,1\} \qquad i = 1,2,\ldots,N \tag{11}$$

$$x_{ij} \ge 0 \qquad i = 1,2,\ldots N, \quad j = 1,2,\ldots,M \tag{12}$$

where

F_i	fixed cost of operating a facility at location i
C_{ij}	cost to satisfy customer j's demand from location i
S_i	capacity of potential location i
D_j	demand for customer j
x_{ij}	demand for customer j shipped from location i
y_i	$\{0,1\}$ indicator where $y_i=1$ implies a facility is located at location i

Specifically, fixed costs (F_i), variable costs (C_{ij}), location capacities (S_i), and customer demands (D_j) are the given input data for the problem and the demand allocation (x_{ij}) and facility locations (y_i) are the decision variables for the problem.

5.2 Solution Representations

Tabu search, simulated annealing and genetic algorithms work on abstract representations of the problem's solution and use the representation's structure to define "moves" or "operators." In the following subsections, we introduce three potential representations for CFLP: a 0-1 vector, a permutation vector, and a matrix. For ease of understanding, we refer to the sample CFLP shown in Figure 8, which consists of five potential locations and ten customers.

	Fixed Costs	1	2	3	4	5	6	7	8	9	10
A	$300	$2	$4	$5	$10	$6	$3	$8	$2	$8	$4
B	$210	$4	$5	$4	$11	$9	$11	$9	$12	$6	$11
C	$150	$12	$2	$3	$4	$10	$2	$7	$11	$ 10	$10
D	$200	$10	$8	$7	$2	$4	$12	$4	$5	$2	$2
E	$275	$6	$7	$12	$3	$6	$4	$5	$3	$4	$7
F	$325	$7	$10	$6	$5	$5	$6	$3	$2	$3	$5
Demand		10	30	10	40	20	10	40	20	10	30

Figure 8. Sample Capacitated Facilities Location Problem (CFLP)

5.2.1 0-1 Vector Representation

The objective in CFLP is to determine the set of locations to operate. The 0-1 vector representation embodies this concept directly. The vector consists of n elements, where each element represents the operational status of a potential location. This is in fact the same solution representation we saw in Chapter 2 for the uncapacitated facilities location problem. For the problem in Figure 8, $\mathbf{s} = \{y_1,\ldots,y_6\}$ where $y_i \in \{0,1\}$, i =1, ..., 6, can represent all possible solutions for this problem.

Associated with each representation is a cost function, $\mathrm{C}:\mathbf{s} \to \Re$, that performs a translation. For the 0-1 vector representation, the translation function consists of first solving the associated transportation problem to determine the proper allocation of customer demand to open locations (i.e., the x_{ij}). Given this set of allocations, the cost is then computed with

$$\mathrm{C}(\mathbf{s}) = \sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}x_{ij} + \sum_{i=1}^{N} F_i y_i$$

5.2.2 Permutation Vector Representation

A 0-1 vector representation is a natural representation for CFLP but is computationally expensive due to the need to solve transportation problems. An alternative representation suggested in Vignaux and Michalewicz (1991) is to number each x_{ij} cell in the problem with a unique number in the range (1..i*j). A permutation vector then represents an ordering of these cell numbers and a specific ordering determines how the x_{ij} will be allocated.

The cost function that translates the permutation vector first determines the allocation for the x_{ij}. It then determines which locations have at least one positive shipment and finally computes the associated cost. To find the allocation for the x_{ij}, the function begins with the first element in the permutation vector. After decoding the corresponding x_{ij} cell, it is assigned the minimum of the corresponding customer's remaining demand or the corresponding location's remaining capacity. The function iterates over each element of the permutation vector in order until all demand has been allocated. Using the values of the x_{ij} we can determine the values for the y_i, and consequently the solution's cost.

5.2.3 A Matrix Representation

Whereas the 0-1 vector and permutation vector representations required some form of translation to get the solution their representation implies, a matrix representation consists of an actual feasible solution representing the allocation of customer demand to potential locations. Since the representation is the solution itself, the cost of the solution can be computed directly.

5.3 A Tabu Search Heuristic for CFLP

In Chapter 2, we defined a tabu search heuristic as the following set of elements:

$$\text{TS} = \{ \mathscr{R}, \mathscr{N}(), \mathscr{C}(), \mathscr{A}(), \mathscr{T}(), \mathscr{P} \}$$

Where

$\mathscr{R}$	is the solution representation structure
$\mathscr{N}()$	is the neighborhood structure;
$\mathscr{C}()$	is the cost function
$\mathscr{A}()$	is the aspiration criterion function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

In the previous section we discussed the three representations and their associated cost functions. We now describe the remaining elements and the associated parameters.

5.3.1 Neighborhood Structure

Neighborhood structures are closely linked to the problem representation. For each representation previously described, we now define their associated neighborhood structure and their size.

For a 0-1 vector solution, a neighbor is defined as a solution where the status of a single location has changed. Thus, for a problem with n potential locations, a solution has a neighborhood size of n. In our implementation, the entire neighborhood is evaluated at each iteration of the algorithm.

For the permutation vector solution, a neighbor is defined as a solution where two of the elements in the vector have swapped positions. Thus, for a problem with n potential locations and m customers, a solution has a neighborhood size of $(n^*m)^*((n^*m)\text{-}1)$. Since this is a large neighborhood, our implementation only evaluates a subset of this neighborhood at each iteration of the algorithm. The

subset consists of swapping each element of the vector once with a randomly selected second element. This results in a neighborhood of size (n^*m).

For the matrix solution representation, a neighbor is defined as a solution where a single allocation of customer demand is moved from its original supplier to another supplier with some positive capacity. The size of this neighborhood is variable depending on the number of positive allocations in the incumbent solution, but it is certainly large. Again, for this representation we only evaluate a subset of the total neighborhood at each iteration of the procedure. The subset consists of randomly selecting an existing allocation of demand for each of m customers, and reallocating as much as possible to a different, randomly selected supplier with some positive remaining capacity. Since only one reallocation is made per customer, the resulting neighborhood is of size m.

5.3.2 Aspiration Criteria

In our implementation we use Glover's original aspiration criteria: a move that is currently "tabu" is only accepted if it results in a neighboring solution whose cost function value is better than the best one so far.

5.3.3 Terminating Condition

The tabu search procedure is terminated when a certain number of iterations have been completed. Two parameters are defined: long-term memory iterations (LTM) and short-term memory iteration (STM). Each long-term memory iteration consists of a restart with a randomly generated solution and an empty tabu list. For each LTM, a number of iterations of the normal tabu search algorithm are executed.

5.3.4 Parameters

The parameter set of our implementation of tabu search consists of three elements:

$\mathscr{P}$ = {LTM, STM, TLSIZE}

LTM: number of long-term memory iterations (restarts).

STM: number of short-term memory iterations.

TLSIZE: tabu list size.

LTM and STM are described in the previous section. For TLSIZE, we use a random tabu list size as in Taillard (1991). TLSIZE represents the maximum size allowed for the tabu list size at any time.

5.4 A Simulated Annealing Heuristic for CFLP

In Chapter 2, we defined a simulated annealing algorithm as the following set of elements:

SA = { $\mathscr{R}$, $\mathscr{N}()$, $\mathscr{C}()$, $\mathscr{S}()$, $\mathscr{T}()$, $\mathscr{P}$ }

where

$\mathscr{R}$	is the solution representation structure
$\mathscr{N}()$	is the neighborhood structure;
$\mathscr{C}()$	is the cost function
$\mathscr{S}()$	is the cooling schedule
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

Solution representations and cost functions were described in §5.2. We now describe the remaining elements.

5.4.1 Neighborhood Structure

The neighborhood structures for simulated annealing are the same as those for tabu search described in §5.3.1. However, in simulated annealing, instead of evaluating all or part of an incumbent solution's neighborhood, we only evaluate

a randomly selected neighbor. For the 0-1 vector representation, we randomly select a potential location and change its status. If the resulting solution is not feasible due to capacity constraints, it is ignored and a new neighbor randomly selected. For the permutation vector representation, we randomly select two elements in the vector and swap their positions. For the matrix representation, we randomly select a customer. For this customer, we randomly select an existing positive allocation of demand to some supplier. We then randomly select a different supplier with some positive capacity and reallocate as much of the demand as possible from the old to the new supplier. In all cases, the result is a new solution with an associated cost.

5.4.2 Cooling Schedule

The cooling schedule consists of selecting an initial temperature, a probability function, and a temperature adjustment function. The initial temperature is selected so that the resulting probability of accepting non-improving solutions is 95%. To find this initial temperature, we take the difference in cost function value between the incumbent solution and the first 100 non-improving solutions (the first 20 for the 0-1 vector representation) and get the average, $\overline{C}$. Using this average we compute the required temperature parameter to achieve the 95% acceptance rate. Since we use the standard probability function $e^{-\Delta/T}$, then the initial temperature parameter $T = -\overline{C}/\ln(.95)$. Finally, the temperature adjustment function is a simple decay function $T^{*} = \alpha T, \quad (\alpha < 1)$. The function is applied at the end of each epoch. The length of an epoch is controlled by two parameters: ACCEPT and FACTOR. An epoch

consists of evaluating enough solutions such that at least some number of solutions are accepted (improving or non-improving). This number of solutions is controlled by the ACCEPT parameter. Since the rate of at which solutions are accepted decreases as the temperature parameter is lowered, the total number of solutions evaluated should be limited to prevent lengthy epochs at lower temperatures. The maximum number of solutions that may be evaluated per epoch is a factor of the ACCEPT parameter controlled by the FACTOR parameter. Thus, an epoch will end when ACCEPT solutions have been accepted or FACTOR*ACCEPT solutions have been evaluated, whichever comes first.

5.4.3 Terminating Condition

The algorithm will terminate whenever the system is deemed frozen. In our implementation, frozen is defined as an acceptance rate of less than 1% during any one epoch.

5.4.4 Parameters

The parameter set of our implementation of simulated annealing consists of three elements:

$\mathscr{P}$ = {RATE, ACCEPT, FACTOR}

RATE: the α parameter in the temperature adjustment function.

ACCEPT: the minimum number of solutions that must be accepted before terminating an epoch.

FACTOR: the factor used in determining the maximum number of solutions that may be evaluated per epoch.

5.5 A Genetic Algorithm Heuristic for CFLP

In Chapter 2, we defined a genetic algorithm as the following set of elements:

GA = { $\mathscr{R}, \mathscr{F}(), \mathscr{X}(), \mathscr{M}(), \mathscr{T}(), \mathscr{P}$}

Where

$\mathscr{R}$	is a solution representation
$\mathscr{F}()$	is the fitness function
$\mathscr{X}()$	is the crossover operation function
$\mathscr{M}()$	is the mutation operation function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

Solution representations are discussed in §5.2. The remaining elements are discussed in the following sections.

5.5.1 Fitness Function

As discussed in Chapter 2, the fitness function assigns a "goodness" value to each chromosome or solution at each generation. In our case, the goodness value is represented by the objective function value, which is the total cost of the solution. Using this value, we assign to each chromosome a probability of selection as a parent for the next generation. Our probability assignment function is taken from Murata, et al. (1996):

$$P_S(\mathbf{x}_i) = \frac{\left[\hat{f}(\Psi) - f(\mathbf{x}_i)\right]^2}{\sum_{x_j \in \Psi} \left[\hat{f}(\Psi) - f(\mathbf{x}_j)\right]^2}$$

where Ψ is the set of chromosomes/solutions in the current generation, $f(\mathbf{x}_i)$ is the fitness value (objective function value) for solution $\mathbf{x}_i$, and $\hat{f}(\Psi)$ is the worst fitness value in the current generation.

The probability of selection computed above is actually used in selecting only the first parent for a crossover operation. The second parent is selected at

random with all chromosomes having an equal probability of selection. This selection process is described in Holland (1975). In addition, we use a biased selection process where the two best chromosomes are copied to the next generation intact.

5.5.2 Crossover Operators

The crossover operation is applied probabilistically to a set of two parent from which two offspring are generated for the new generation. With probability XOVER, the crossover operator is applied to the parents to generate the first offspring. With complementary probability, the offspring is a copy of the first parent. With probability XOVER, the crossover operator is applied to the parents (in reverse order) to generate the second offspring. With complementary probability, the offspring is a copy of the second parent. The crossover operators are defined appropriately for each solution representation.

For the 0-1 vector representation, we use a two-point crossover operator. The two crossover points are selected at random. If applied, the first offspring receives the portion of the solution vector between the two points from the first parent, and the head and tail portions from the second parent. The second offspring receives the middle portion from the second parent, and the head and tail portions from the first parent. Since this kind of crossover operator can lead to infeasible solutions, a repair operation must also be added. The repair operation is applied whenever the resulting offspring has insufficient capacity to meet demand and consists of opening additional, randomly selected locations until enough capacity is available.

For the permutation vector, we also use a two-point crossover operator. The first crossover point is selected from the first $N+M$ elements, where N is the number of locations in the problem, and M the number of customers. This is done to ensure that at each crossover operation, some of the early elements in the permutation are changed, which are the most significant with this kind of representation. The second crossover point is selected at random and to the right of the first point. The first offspring receives the elements between the crossover points from the first parent. The remaining elements are filled from head to tail in the same order as they appear in the second parent (i.e., the elements not copied from the first parent). The operation is the same for the second offspring with the roles of the parents reversed. The resulting offspring are always feasible.

For the matrix representation we use a single crossover point operation. Recall the matrix representation is a N x M tableau solution to the problem. The crossover point is randomly selected as one of the M customers. The first offspring will receive the columns to the left of the crossover point from the first parent matrix and the remaining columns from the second parent. The operation is the same for the second offspring with the roles of the parents reversed. As in the 0-1 vector representation, there is no guarantee the resulting offspring solutions will be feasible, requiring a repair operation. The repair operation is applied to each row in the matrix (i.e., each potential location), whenever the assigned allocations exceed the corresponding capacity. The repair operation consists of randomly selecting allocations and moving them to other, randomly

selected locations with some positive capacity. This is done until the total demand on a location is less than its capacity, and for each location until the solution is feasible.

For the matrix representation, we also tried to use the matrix crossover operator of Vignaux and Michalewicz (1991) described in Chapter 3. However, results were inferior to those with the operator described above; we therefore abandoned this approach.

5.5.3 Mutation Operators

The mutation operator is applied probabilistically. In the case of the matrix representation, the mutation is applied probabilistically to each gene (column) of the chromosome. For the other two representations, the operator is applied probabilistically to each chromosome in the population.

For the 0-1 vector representation, the mutation operation is applied to each chromosome with MUTATE probability. The operation consists of selecting a random element in the vector and switching its status. Notice that when a location is closed, the resulting offspring may be infeasible. This situation would be corrected during the repair operation as described in the previous section.

For the permutation vector representation, the mutation operator is also applied to each chromosome with MUTATE probability. The operation consists of randomly selecting one of the first $M + N$ elements, and swapping it with another, randomly selected element. The results are always feasible.

For the matrix representation, the mutation operator is applied to each gene (column) in each chromosome with MUTATE probability. The mutation consists

of selecting a random allocation for the customer, and reallocating from the current supplier to a different, randomly selected supplier with some positive capacity. The smaller of the existing allocation or the available capacity in the new supplier is reallocated. The results are always feasible.

5.5.4 Terminating Condition

The genetic algorithm is terminated when a fixed number of generations, GENS, have elapsed or if the algorithm converges to a single solution (i.e., all of the chromosomes in a generation represent the same solution). Convergence only tends to happen when the mutation rate MUTATE is small (e.g., 1%).

5.5.5 Parameters

The parameter set of our implementation of a genetic algorithm consists of four elements:

$$\mathscr{P} = \{\text{POP, GENS, XOVER, MUTATE}\}$$

POP = the population size of each generation (i.e., number of chromosomes).

GENS = the number of generations before terminating the algorithm.

XOVER = the probability of applying the crossover operation.

MUTATE = the probability of applying the mutation operation.

5.6 Benchmark Test Problems

A set of 20 medium-sized test problems was selected for the purposes of representation selection, parameter effects experimentation, and benchmarking. The 20 test problems are based on a set of uncapacitated location problems found in Khuen and Hamburger (1963) and later modified as capacitated location

Table 8. Benchmark Test Problems (CFLP)

Set	Size	Capacity	Fixed Cost
I	16x50	10000	7500/12500/17500/2500
II	16x50	15000	7500/12500/17500/2500
III	25x50	5000	7500/12500/17500/2500
IV	25x50	15000	7500/12500/17500/2500
V	25x50	58268	7500/12500/17500/2500

problems by Sa (1969) and later used by Ellwein (1971), Khumawala (1974), and Akinc and Khumawala (1977). The set contains 8 problems with 16 locations and 50 customers, and 12 problems with 25 locations and 50 customers. The problems are summarized in Table 8. The capacity indicated is the capacity assigned to each potential location. For each size-capacity combination, there are four levels of fixed cost per potential location. Optimal solutions for each of these problems were obtained from a branch and bound algorithm.

5.7 Representation Selection

We conducted a small experiment in order to determine which of the three solution representations is best suited for capacitated facility location problems.

Table 9. Parameters for Representation Selection (CFLP)

	0-1 Vector	Permutation	Matrix
TS-CFLP			
LTM	5	5	5
STM	20	1500	2000
TLSIZE	7	7	7
SA-CFLP			
RATE	.8	.9	.85
ACCEPT	15	1000	1000
FACTOR	2	2	3
GA-CFLP			
POP	30	40	40
GENS	75	1500	1500
XOVER	.85	.80	.80
MUTATE	.05	.05	.05

In this experiment, each of the three general heuristics was used to solve each of the 20 benchmark test problems using each of the three solution representations, resulting in an experiment with 180 cells. Due to this number of cells, we completed five replications per cell. The parameters used for each of the nine heuristic-representation combinations are shown in Table 9.

In order to determine which representation would be most appropriate, we first ranked the performance of the heuristics with each representation. The results of these rankings are shown in Table 10. The raw results are shown in Tables 95, 96 and 97 in Appendix A. The rankings clearly show that the 0-1 vector representation dominates the other two representations. In particular, for medium-sized problems such as those in our benchmarking set, the 0-1 vector

Table 10. Representation Performance Ranks (CFLP)

Prob.	TS Solution Ranks			SA Solution Ranks			GA Solution Ranks		
#	V	P	T	V	P	T	V	P	T
1	1	3	2	1	3	2	1	3	2
2	1	3	2	1	3	2	1	3	2
3	1	3	2	1	3	2	1	3	2
4	1	2	3	1	2	3	1	3	2
5	1	2	3	1	3	2	1	3	2
6	1	3	2	1	3	2	1	3	2
7	1	2	3	1	2	3	1	3	2
8	1	2	3	1	2	3	1	3	2
9	1	3	2	1	3	2	1	3	2
10	1	3	2	1	3	2	1	3	2
11	1	2	3	1	3	2	1	3	2
12	1	2	3	1	3	2	1	3	2
13	1	3	2	1	3	2	1	3	2
14	1	3	2	1	3	2	1	3	2
15	1	3	2	1	3	2	1	3	2
16	1	2	3	1	3	2	1	3	2
17	1	3	2	1	2	3	1	3	2
18	1	3	2	1	2	3	1	3	2
19	1	3	2	1	2	3	1	3	2
20	1	3	2	1	2	3	1	3	2

representation is most appropriate. However, since we are also interested in applying these general heuristics to larger sized problems, computation time is also a key determinant of appropriateness. In Figures 9, 10 and 11 we show the time performance of each representation for tabu search, simulated annealing, and the genetic algorithm respectively. The problems along the x-axis are in the same order as in Table 10, sorted by their size, then by their capacity tightness, and lastly by their fixed cost. Notice that the time performance for the 0-1 vector representation (V) appears to rise sharply as the size and complexity of the problem increase, whereas the rise for the other two representations is not as pronounced. In particular, the matrix representation (T) does not seem to be heavily affected by the size or complexity of the problem.

At this point, the utility of a 0-1 vector representation for larger sized problems is suspect. We conducted the same experiment above for an additional test problem from our set of larger sized problems. This problem consists of 25 locations and 200 customers. This is the smallest of the larger sized problems. The results are shown in Figures 12, 13, and 14. These figures clearly show the performance of the 0-1 vector representation is unsuitable for larger sized problems. Therefore, we are left with a compromise solution representation that balances solution quality and time performance: the matrix/tableau representation. These results answer the first research question with respect to capacitated facility location problems.

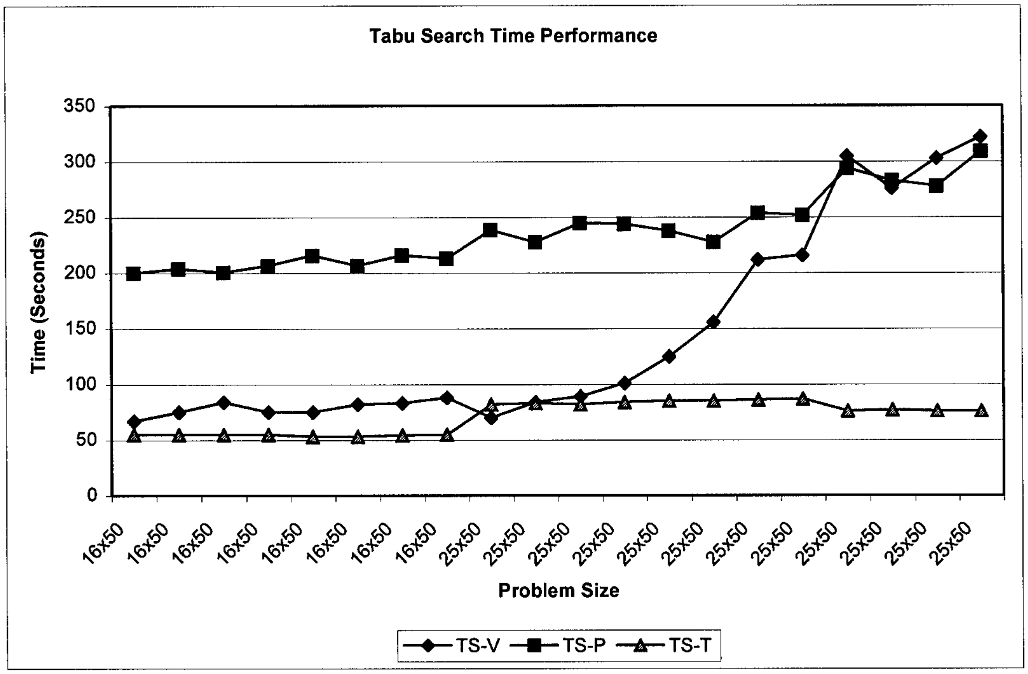

Figure 9. TS Time Performance (CFLP)

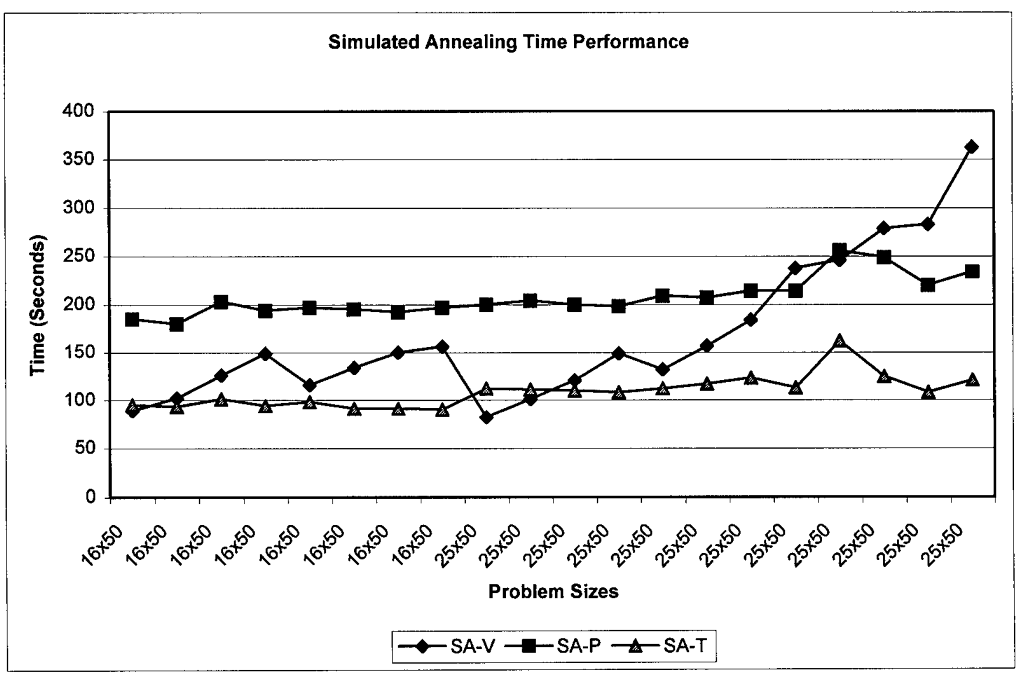

Figure 10. SA Time Performance (CFLP)

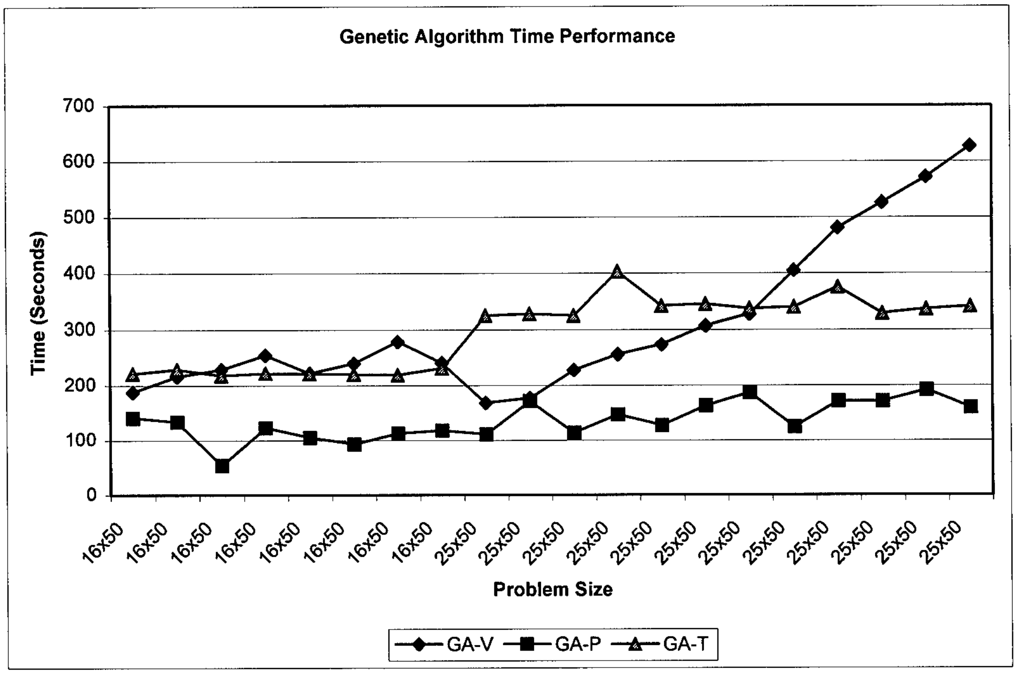

Figure 11. GA Time Performance (CFLP)

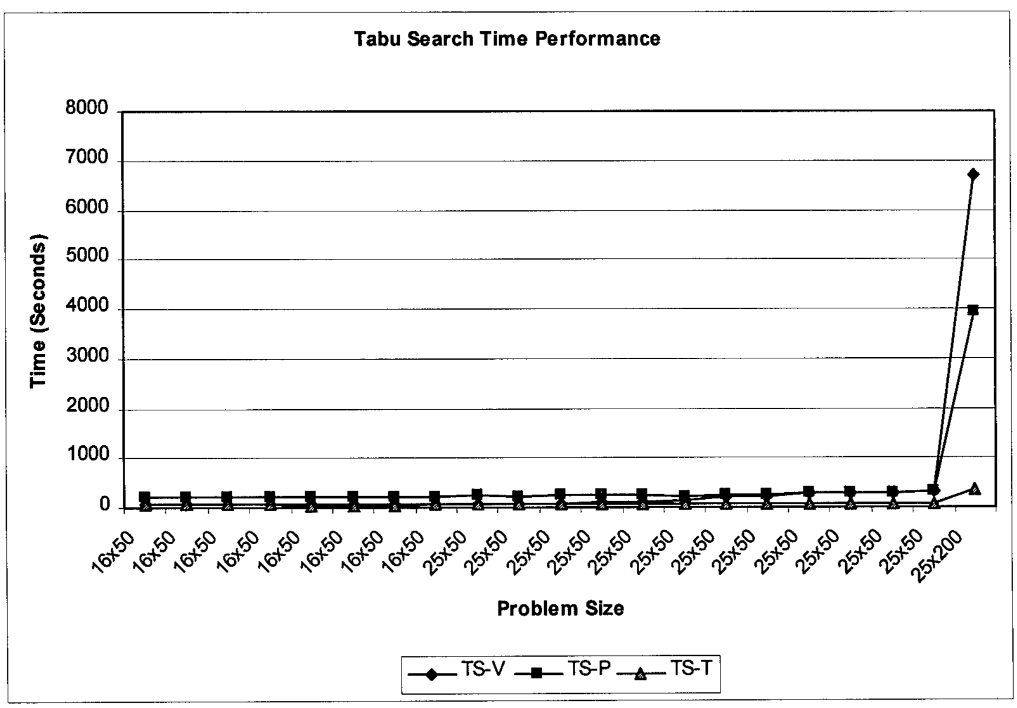

Figure 12. TS Time Performance with Larger Problem (CFLP)

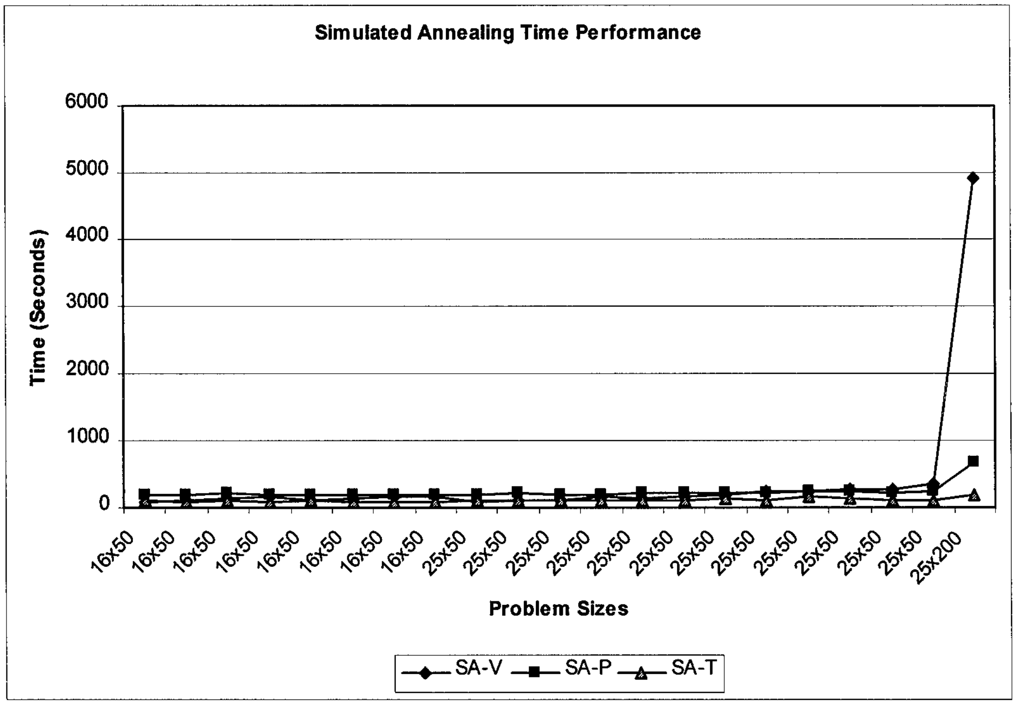

Figure 13. SA Time Performance with Larger Problem (CFLP)

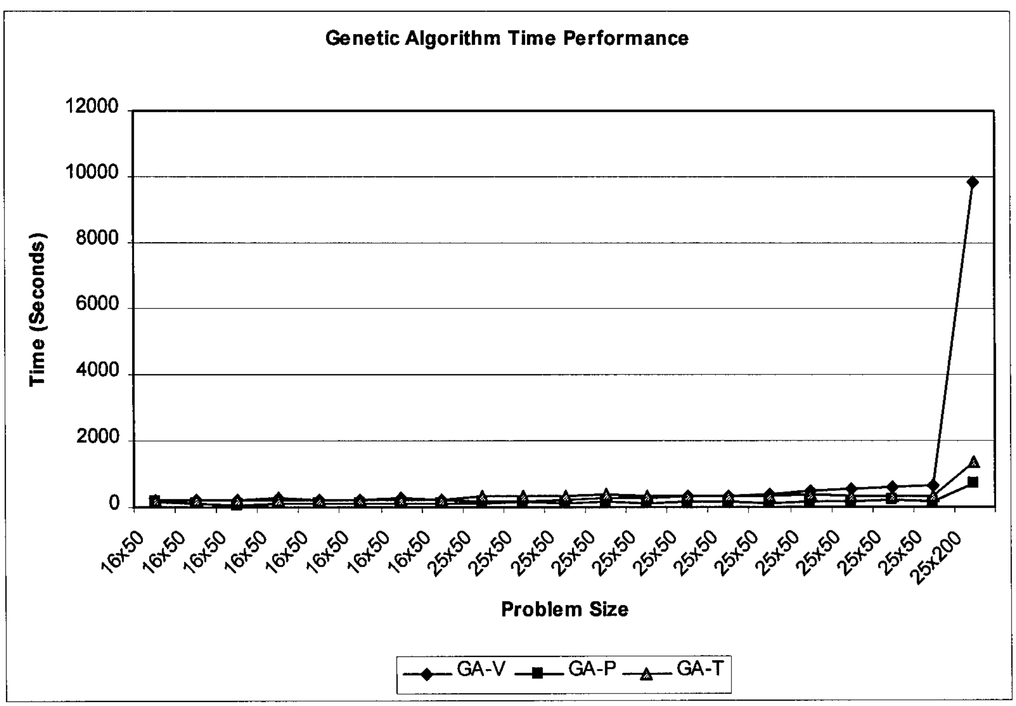

Figure 14. GA Time Performance with Larger Problem (CFLP)

5.8 Parameter Effects

From the literature, it is understood that the performance of tabu search (TS), simulated annealing (SA) and genetic algorithms (GA) depends on the parameter settings. Accordingly, we conducted an experiment with each of the general heuristics to determine the effect of various parameter settings on solution quality and computation time.

For tabu search, we experimented with three levels for the number of restarts (LTM), five levels for the number of iterations per restart (STM), and three levels of maximum tabu list size (TLSIZE). The experimental levels are summarized in Table 11. The three levels of maximum tabu list size correspond to 10%, 15%, and 25% of the neighborhood size (i.e., number of customers).

Table 11. TS Parameters Experimental Levels (CFLP)

Parameter	Experimental Levels				
LTM	1	5	10		
STM	1000	1500	2000	2500	3000
TLSIZE	5	7	12		

For simulated annealing, we experimented with five levels for the RATE parameter, four levels for the ACCEPT parameter, and three levels for the FACTOR parameter. The experimental levels are summarized in Table 12.

Table 12. SA Parameters Experimental Levels (CFLP)

Parameter	Experimental Levels				
RATE	0.75	0.80	0.85	0.90	0.95
ACCEPT	1000	1500	2000	2500	
FACTOR	1	2	3		

For genetic algorithms, we experimented with three levels for the POP, GENS, and XOVER parameters. The MUTATE parameter was set at two levels. The experimental levels are summarized in Table 13.

Table 13. GA Parameters Experimental Levels (CFLP)

Parameter	Experimental Levels		
POP	20	30	40
GENS	1000	1500	2000
XOVER	0.85	0.95	1.0
MUTATE	0.05	0.10	

To conduct these experiments, one of the test problems from the benchmark set was selected as a test bed. Looking back at the results from the representation selection experiments, we selected a problem that was generally difficult to solve for all of the general heuristics. Difficulty is measured by the resulting optimality gap. For tabu search and simulated annealing, this problem had the worst optimality gap performance at 11.33% and 14.32% respectively. The genetic algorithm also had difficulty with this problem resulting in an optimality gap of 32.29%, although this was not the worst performance. The problem selected has 25 potential locations, 50 customers, a limited capacity of 5000 per location, and a fixed cost of $25,000 per location.

The tabu search experiment has 45 cells, the simulated annealing experiment has 60 cells and the genetic algorithm experiment has 54 cells. Considering the total number of experimental cells, we limited each cell to 10 replications. In the following sections, we summarize the effects of each parameter. Tables 14 and 15 summarize the results of the Kruskal-Wallis H tests

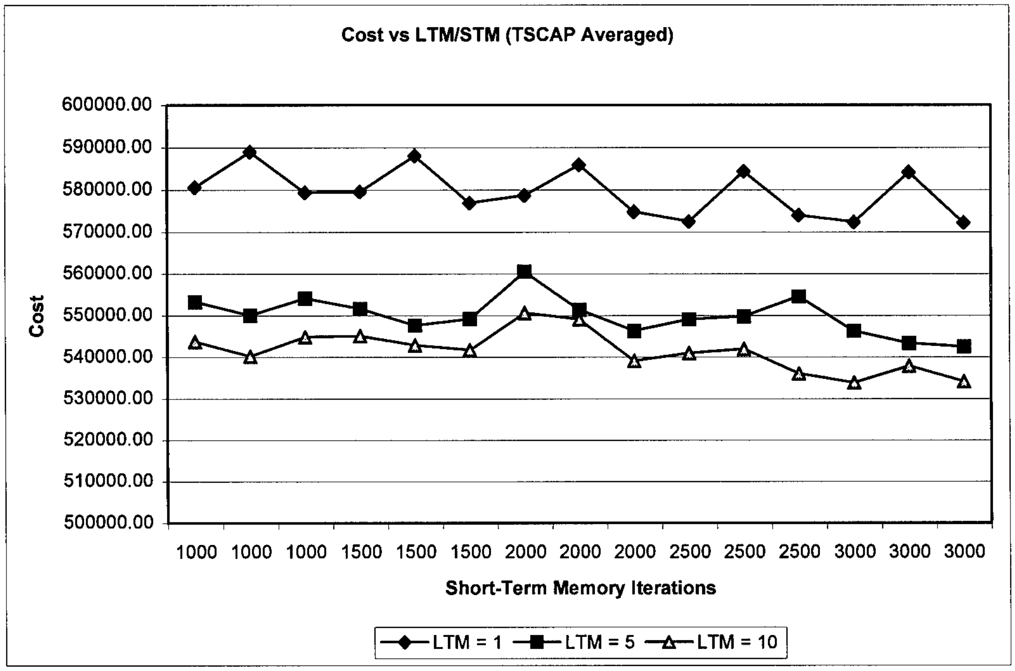

Figure 15. TS Parameters Effect on Cost (CFLP)

and tables 16 and 17 summarize Spearman's correlation coefficients. These results answer the second research question with respect to CFLP.

5.8.1 Tabu Search LTM Parameter

The effect of increasing the number long-term memory passes (number of restarts), is clear from Figure 15. In this figure, each line represents a level of the LTM parameter, and each point the averaged cost of ten replications. Going from a single pass to five passes has a large impact for all test problems, reducing the overall solution costs significantly. Going from five passes to ten passes also reduces overall costs, although the effect is not as dramatic. The Kruskal-Wallis H test shows the LTM parameter has a significant effect on solution quality (see Table 14).

With respect to computation time, the effect of increasing the number of passes is as expected and shown in Figure 16. Each line in Figure 16

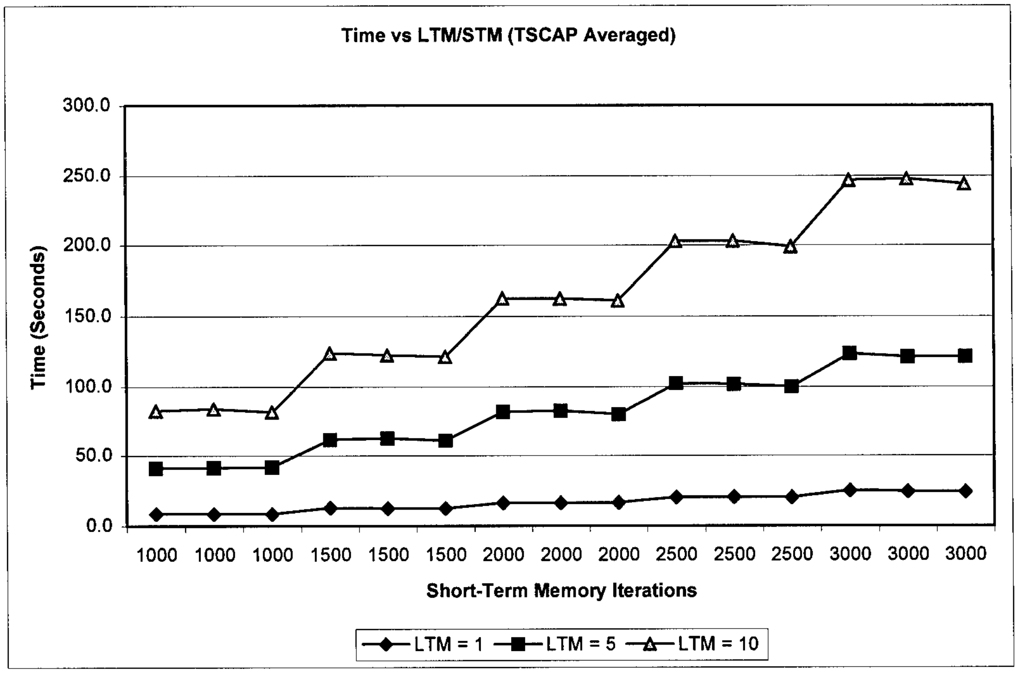

Figure 16. TS Parameters Effect on Time (CFLP)

represents each level of the LTM parameter. More passes clearly require more computation time. This significant effect is also confirmed by the Kruskal-Wallis H test (see Table 15).

5.8.2 Tabu Search STM Parameter

The effects of changing the STM parameter are also visible in Figure 15. As the number of iterations increases (shown on the x-axis), the average cost goes down for each of the three levels of the LTM parameter. The decrease rate is not very steep at all, and is almost flat for a single pass of the long-term memory. The K-W test for this parameter is significant (p = .0304) and a Spearman correlation coefficient of -.1272 (p = .007) confirms the slight trend.

With respect to computation time, Figure 16 shows that more iterations require more computation time as one would expect. However, there is an interaction between the LTM and STM parameters. The rate at which

Table 14. Kruskal-Wallis H Test of Heuristic Parameters Effect on Solution Quality (CFLP)

Parameter	Chi-Square	D.F.	Significance
LTM	224.2147	2	.0000
STM	10.6836	4	.0304
TLSIZE	2.7052	2	.2586
RATE	72.4681	4	.0000
ACCEPT	28.4350	3	.0000
FACTOR	29.5130	2	.0000
POP	2.3462	2	.3094
GENS	16.9384	2	.0002
XOVER	1.3125	2	.5188
MUTATE	376.3415	1	.0000

Table 15. Kruskal-Wallis H Test of Heuristic Parameters Effect on Computation Time (CFLP)

Parameters	Chi-Square	D.F.	Significance
LTM	359.7749	2	.0000
STM	81.2943	4	.0000
TLSIZE	.2126	2	.8992
RATE	359.6451	4	.0000
ACCEPT	112.9145	3	.0000
FACTOR	99.3806	2	.0000
POP	223.5026	2	.0000
GENS	262.6808	2	.0000
XOVER	.2067	2	.9018
MUTATE	17.1998	1	.0000

computation time increases for each level of STM increases as the LTM parameter is also increased. Considering both effects on solution quality and computation time, the obvious implication is that increasing the STM parameter is not as effective as increasing the LTM parameter given some limit on computation resources.

5.8.3 Tabu Search TLSIZE Parameter

Looking at Figure 15, notice that for each level of short-term memory iterations, there are three points. These points correspond to each level of TLSIZE. Unfortunately, the effects of this parameter are not entirely clear. For the line corresponding to a single pass of the long-term memory, it is obvious that

the middle level of TLSIZE, 7, gives the worst performance. The other two levels result in comparable solution quality. For the other two levels of LTM, the results are more complicated. For lower levels of STM, the middle level of TLSIZE, 7, gives the best performance. However, as the level of STM increases, the effect reverses. Interestingly, many applications of tabu search found in the literature use a tabu list size of 7. However, the present results seem to indicate that the effect of this parameter is very sensitive to the settings for the LTM and STM parameters. The K-W test ($p = .2586$) and the Spearman correlation coefficient ($p = .173$) show that this parameter does not have a significant effect on solution quality.

With respect to time, Figure 16 shows that TLSIZE has very little impact on computation time. The K-W test ($p = .8992$) and Spearman correlation coefficient ($p = .665$) confirm this observation. However, there is a slight yet consistent trend showing a decrease in computation time as the TLSIZE parameter increases. Presumably, this effect comes from the fact that as more moves are declared "tabu," fewer solutions must be evaluated.

5.8.4 Tabu Search Selected Parameters

Based on the previous results, we set the parameters for further experimentation to the following levels: LTM = 10, STM = 3000, and TLSIZE = 5.

5.8.5 Simulated Annealing RATE Parameter

Figure 17 summarizes the effects of simulated annealing parameters on solution quality (cost). Each line represents a different experimental level of the RATE parameter. Each point is the average cost of ten replications. Looking at

Table 16. Spearman Correlation Coefficients of Heuristic Parameters vs Solution Quality

Parameter	Coefficient	Significance
LTM	-.6863	.000
STM	-.1272	.007
TLSIZE	-.0644	.173
RATE	-.3420	.000
ACCEPT	-.2130	.000
FACTOR	-.2051	.000
POP	-.0572	.184
GENS	-.1753	.000
XOVER	.0102	.814
MUTATE	.8356	.000

Table 17. Spearman Correlation Coefficient of Heuristic Parameters vs Computation Time

Parameter	Coefficient	Significance
LTM	.8906	.000
STM	.4245	.000
TLSIZE	-.0205	.665
RATE	.7639	.000
ACCEPT	.4314	.000
FACTOR	.4002	.000
POP	.6411	.000
GENS	.6933	.000
XOVER	-.0007	.986
MUTATE	.1786	.000

this figure, a general result is that as the RATE parameter increases, solution quality improves. Unfortunately, this is not a consistent result across levels of the ACCEPT parameter (x-axis). The two highest levels of RATE, 0.90 and 0.95, do result in the best performance consistently. Spearman's correlation coefficient of -.3420 ($p = .000$) confirms the direction and significance of the effect.

Figure 18 summarizes the effects of simulated annealing parameters on computation time. The effects are clear, higher levels of RATE require more computation time due to the slower cooling rate. Notice that the increase in time going from a RATE level of 0.90 to 0.95 is much larger than for the other levels. Again, the K-W test ($p = .000$) and Spearman's correlation coefficient of .7639 ($p = .0000$) confirm these observations.

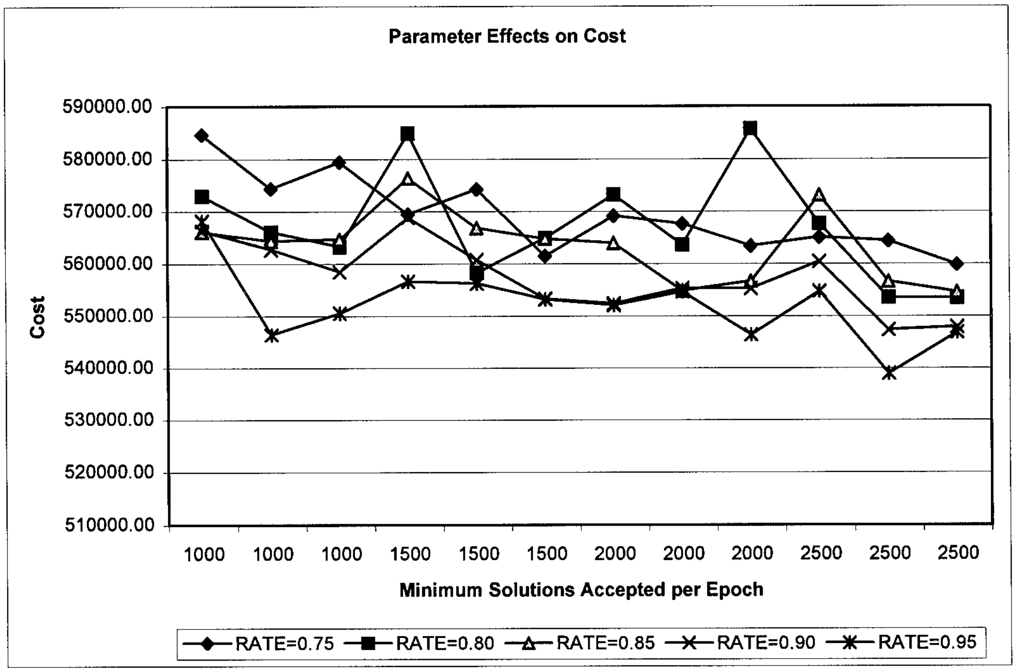

Figure 17. SA Parameters Effect on Cost

5.8.6 Simulated Annealing ACCEPT Parameter

Figure 17 shows the levels of the ACCEPT parameter on the x-axis. The general effect is a slight improvement in solution quality as the ACCEPT parameter increases. However, the effect is not consistent. It is most visible for the RATE levels of 0.75 and 0.90, and most inconsistent for the RATE level of 0.85.

With respect to time, Figure 18 clearly shows that as the ACCEPT parameter increases, total computation time also increases. This is an expected result as more solutions are evaluated for higher levels of ACCEPT.

5.8.7 Simulated Annealing FACTOR Parameter

The effects of FACTOR on solution quality can be seen in Figure 17. Notice on the x-axis each level of the ACCEPT parameter has three points, each corresponding to a level of the FACTOR parameter. Unfortunately, not much can

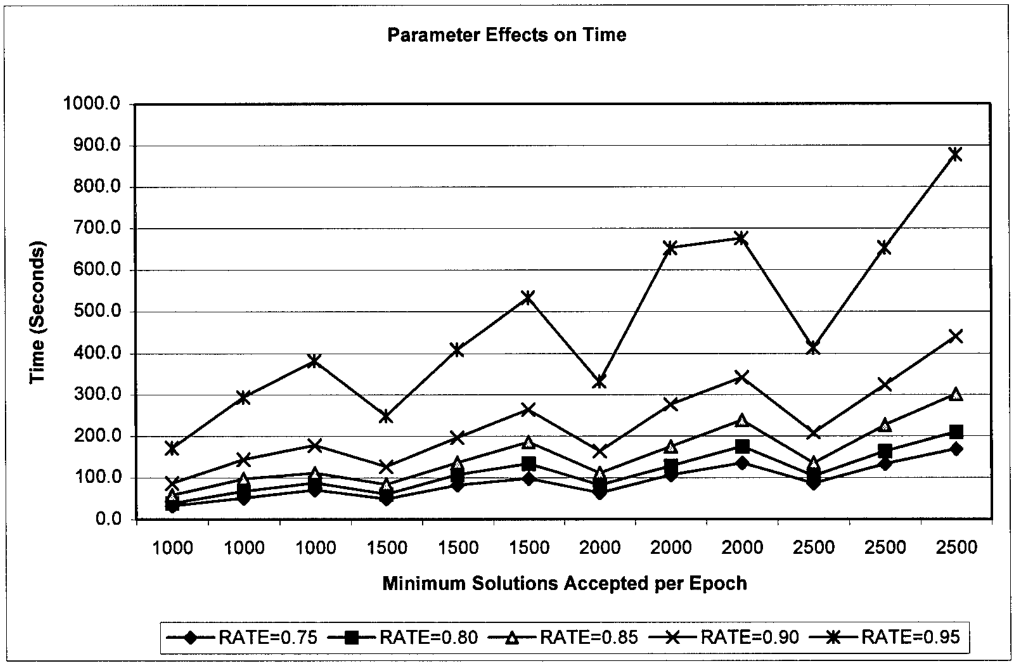

Figure 18. SA Parameters Effect on Time

be said of this parameter. A larger FACTOR level does not necessarily result in superior solutions, even though more solutions are allowed per epoch at the lower temperatures in the cooling schedule.

The effects on computation time, however, are clearly visible in Figure 18. For each RATE-ACCEPT combination, higher levels of FACTOR require more computation time as expected. Notice that the rate of increase goes up as the levels of the RATE parameter also go up.

5.8.8 Simulated Annealing Selected Parameters

Based on the preceding results, parameter settings for further experiments are set at the following levels: RATE = 0.90, ACCEPT = 2500, and FACTOR = 2. These settings achieve good results within reasonable computation times.

5.8.9 Genetic Algorithm POP Parameter

Figure 19 shows the genetic algorithm parameter effects on solution quality (cost). Each line in the graph corresponds to a POP-MUTATE parameter combination. The legend at the bottom identifies each line with a number and a letter. The number refers to the levels of POP, and the letter to the levels of MUTATE: A = 0.05 mutation rate, and B = 0.10 mutation rate. Considering only the POP parameter, the effect would seem to be slight. For the lines with a MUTATE rate of 0.10, the effect is inconsistent. For the lines with a MUTATE rate of 0.05, a higher level of POP results in a better solution, though only slightly. Neither the K-W test ($p = .3094$) nor Spearman's correlation coefficient ($p = .184$) show a significant effect at the .05 level.

Figure 20 shows the effects of parameters on computation time. The results are clear and consistent. Higher levels of POP require more time as expected, due to the larger number of solutions evaluated.

5.8.10 Genetic Algorithm GENS Parameter

Figure 19 shows the effect of the number of generations (GENS) on solution quality. The number of generations is shown on the x-axis. As more generations are allowed, the solution quality improves. However, there is a leveling-off effect and the rate of improvement does not appear to be very large. This is confirmed by the low Spearman's correlation coefficient of -.1753 ($p = .000$).

Figure 20 shows the effects of the GENS parameter on computation time. Again, as expected higher levels of the GENS parameter require more time. Notice that the rate of increase as the POP parameter increases does not

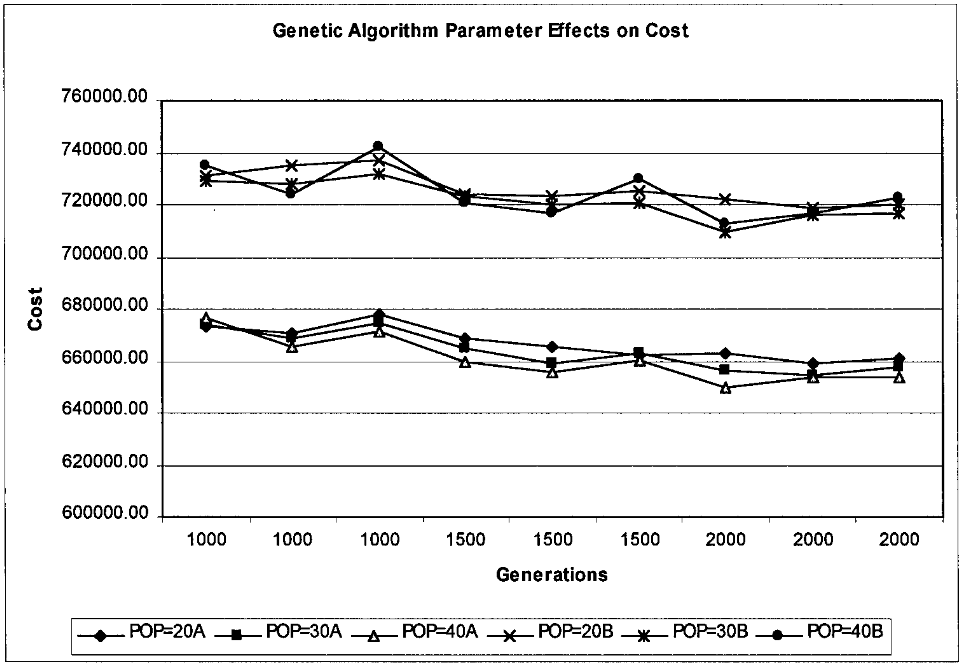

Figure 19. GA Parameters Effect on Cost

change much, unlike the effect seen for tabu search and simulated annealing of having more iterations.

5.8.11 Genetic Algorithm XOVER Parameter

The effects of the XOVER parameter on solution quality can be seen in Figure 19. Notice on the x-axis, each level of the GENS parameter has three points. These points correspond to each level of the XOVER parameter. The only observation that can be made is that the effect of the XOVER parameter on solution quality is sensitive to the settings of the other parameters. The statistical tests show no significant effect on solution quality.

With respect to computation time, the effect can be seen in Figure 20. Generally, the XOVER parameter does not seem to have much effect on computation time, which is confirmed by the statistical tests.

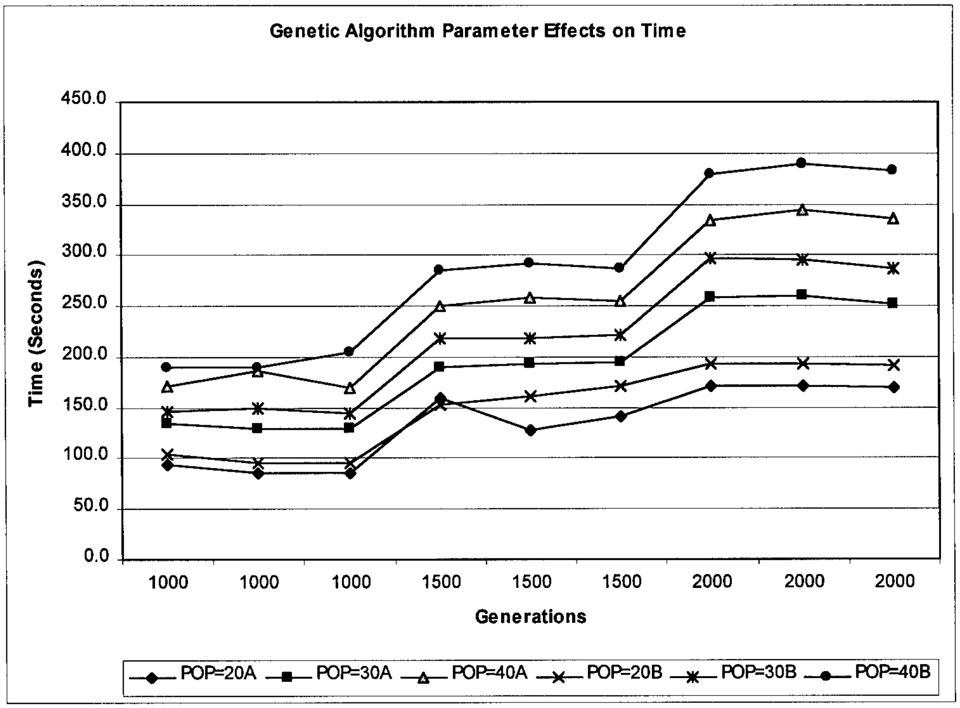

Figure 20. GA Parameters Effect on Time

5.8.12 Genetic Algorithm MUTATE Parameter

The effects of the MUTATE parameter on solution quality are clearly visible in Figure 19. The MUTATE parameter has the most significant effect on solution quality. Specifically, a 5% mutation rate is much better than a 10% mutation rate. A higher mutation rate has a tendency to disrupt the inheritance property of genetic algorithms and this may be the reason for its poor performance. From earlier experiments not reported here, we also know that mutation rates lower than 5%, (e.g., 1%) are also ineffective as they lead to early convergence to a single, poor solution.

With respect to time, the effects can be seen in Figure 20. In general, the higher mutation rate requires slightly more computation time. This is as expected since a higher mutation rate implies more changes to a solution.

5.8.13 Genetic Algorithm Selected Parameters

Based on the preceding results, we selected the following parameter settings for further experimentation with genetic algorithms: POP = 40, GENS = 2000, XOVER = 0.85, and MUTATE = 0.05. These parameters gave the best results.

5.9 Benchmark Test Results

In this section we report the results from benchmarking experiments. We used the 20 benchmark test problems described earlier in Table 8 that were also used for the representation selection experiments. Each of these problems was first solved to optimality using a branch and bound algorithm based on the work of Akinc (1973). We then solved each problem using each of the general heuristics with the matrix representation. Parameter settings were selected based on the results from the previous parameter experiments and are summarized in Table 18. For each problem we completed ten replications using each heuristic. The average results and optimality gaps are reported in Table 19.

Table 18. Parameter Settings for Benchmark Experiments (CFLP)

TS Parameters		SA Parameters		GA Parameters	
LTM	10	RATE	0.90	POP	40
STM	3000	ACCEPT	2500	GENS	2000
TLSIZE	5	FACTOR	2	XOVER	0.85
				MUTATE	0.05

The benchmark experiments show that tabu search has an average gap from optimality of 2.32%, simulated annealing has an average gap from optimality of 3.17%, and the genetic algorithm has an average gap from optimality of 21.28%. The genetic algorithm did not perform well for any of the test cases even though it consistently took the longest time (except for problem CAP201) to finish.

Table 19. CFLP Benchmark Results

	Tabu Search			Simulated Annealing			Genetic Algorithm		
Problem	Solution	Opt Gap	Time	Solution	Opt Gap	Time	Solution	Opt Gap	Time
CAP201	$547,050	3.18%	186	$554,137	4.51%	313	$634,624	19.69%	308
CAP202	$489,730	1.13%	205	$490,341	1.26%	270	$548,589	13.28%	307
CAP203	$447,661	0.66%	236	$448,174	0.78%	266	$482,770	8.56%	307
CAP204	$394,224	0.26%	202	$394,923	0.43%	280	$416,375	5.89%	308
CAP205	$575,541	2.04%	183	$582,134	3.21%	276	$668,237	18.47%	330
CAP206	$515,926	1.05%	182	$519,233	1.69%	264	$575,904	12.79%	306
CAP207	$472,099	0.92%	193	$474,845	1.51%	262	$514,544	10.00%	327
CAP208	$420,935	1.11%	185	$418,568	0.54%	268	$443,881	6.62%	306
CAP209	$424,042	6.73%	271	$421,020	5.97%	322	$583,129	46.77%	456
CAP210	$346,748	3.11%	269	$349,581	3.95%	317	$444,769	32.26%	456
CAP211	$287,122	1.94%	266	$287,172	1.96%	318	$350,226	24.34%	453
CAP212	$206,788	0.52%	277	$206,471	0.36%	314	$238,903	16.13%	456
CAP213	$413,088	4.06%	266	$415,475	4.66%	300	$566,285	42.65%	461
CAP214	$336,213	2.29%	271	$338,265	2.91%	353	$437,747	33.18%	456
CAP215	$276,764	1.23%	267	$276,028	0.96%	299	$340,305	24.47%	457
CAP216	$199,888	0.37%	267	$200,222	0.54%	356	$233,000	17.00%	458
CAP217	$533,818	7.28%	254	$547,332	9.99%	354	$649,615	30.55%	456
CAP218	$420,496	4.10%	260	$441,898	9.40%	313	$500,692	23.96%	453
CAP219	$344,534	3.66%	249	$349,627	5.20%	324	$404,456	21.69%	461
CAP220	$253,076	0.74%	252	$260,006	3.49%	312	$294,771	17.33%	454

These results address the third research question with respect to capacitated facility location problems.

5.10 Computation Time vs. Solution Quality

To determine what kind of relationship exists between computation time and solution quality we again use the data collected during the parameter selection experiments. Tables 15 and 17 indicate that most parameters have a significant impact on computation time. The immediate by-product of these experiments is a set of data points with a range of computation times for the same problem with each heuristic. Thus, we can use Spearman's rank correlation coefficient to determine if there is a relationship between solution time and solution quality.

The correlation coefficients are summarized in Table 20. Note that as expected, all coefficients show a negative correlation, meaning solution quality improves with longer computation time. However, this relationship is significant only for tabu search and simulated annealing. For genetic algorithms, the correlation coefficient is not significant.

Table 20. Spearman's Rank Correlation Coefficients for Computation Time vs. Solution Quality (CFLP)

Heuristic	Spearman's	Significance
Tabu Search	-.6827	.000
Simulated Annealing	-.4505	.000
Genetic Algorithm	-.0274	.525

The best way to appreciate how each of these heuristics performs over time is to see how the heuristics make improvements to the solution over time. In Figure 21, we show the performance over time for each heuristic. We use the same test problem used for the parameters experiment with the heuristic parameters set to the same levels as for benchmarking shown in Table 18.

Looking at Figure 21, it is obvious that tabu search is the most efficient heuristic with respect to time, quickly finding better solutions. The flat area where apparently the best solution is not improved for a long time is an effect of the long term memory parameter (re-starts). Since each re-start begins with a new random solution, it is not until the last few iterations of each re-start that the best solution from previous starts has a chance to be improved. In this case, the best solution from the first or second re-starts is not improved until the last re-start.

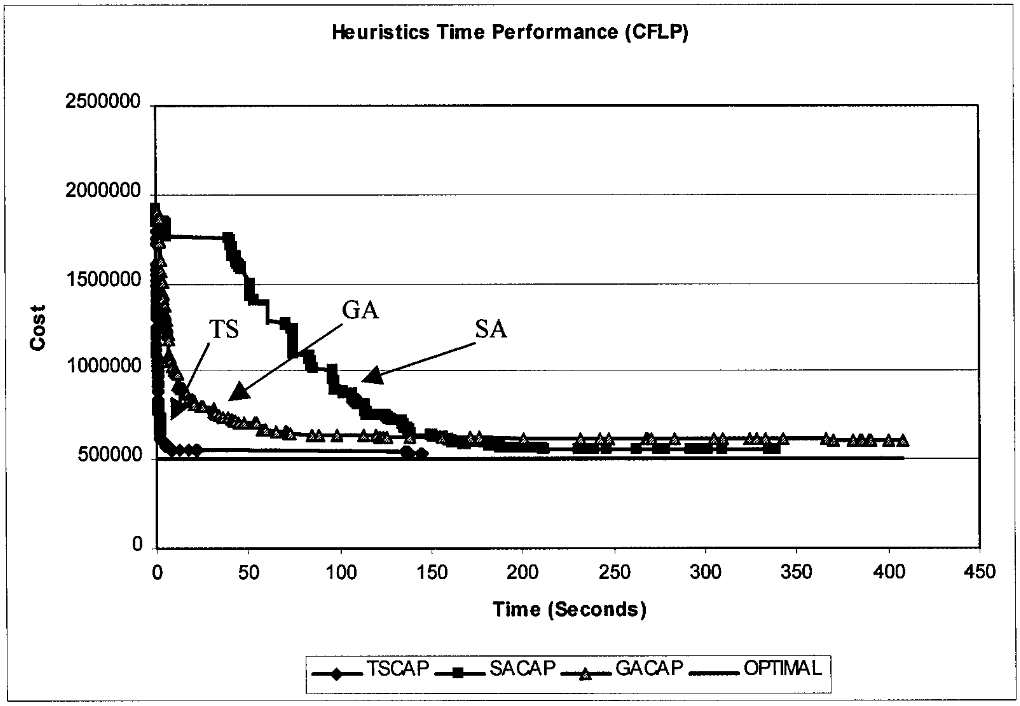

Figure 21. Heuristics Time Performance (CFLP)

The performance of simulated annealing is more deliberate with a slow, but consistent rate of improvement. However, it takes much longer for simulated annealing to reach the same level of performance as tabu search.

Finally, the genetic algorithm appears to be quite efficient at first, but begins to level off too early. Although it continues to improve the solutions over time, its rate of improvement slows down so that it seems it will never actually catch up with tabu search, and is eventually passed by simulated annealing.

5.11 Problem Size vs Computation Time

To explore the relationship between problem size and computation time we use the data collected during the benchmarking experiments. These data consists of 20 problems, 8 of size 16x50 and 12 of size 25x50, each solved ten times with each of the general heuristics. We coded the smaller problems with a

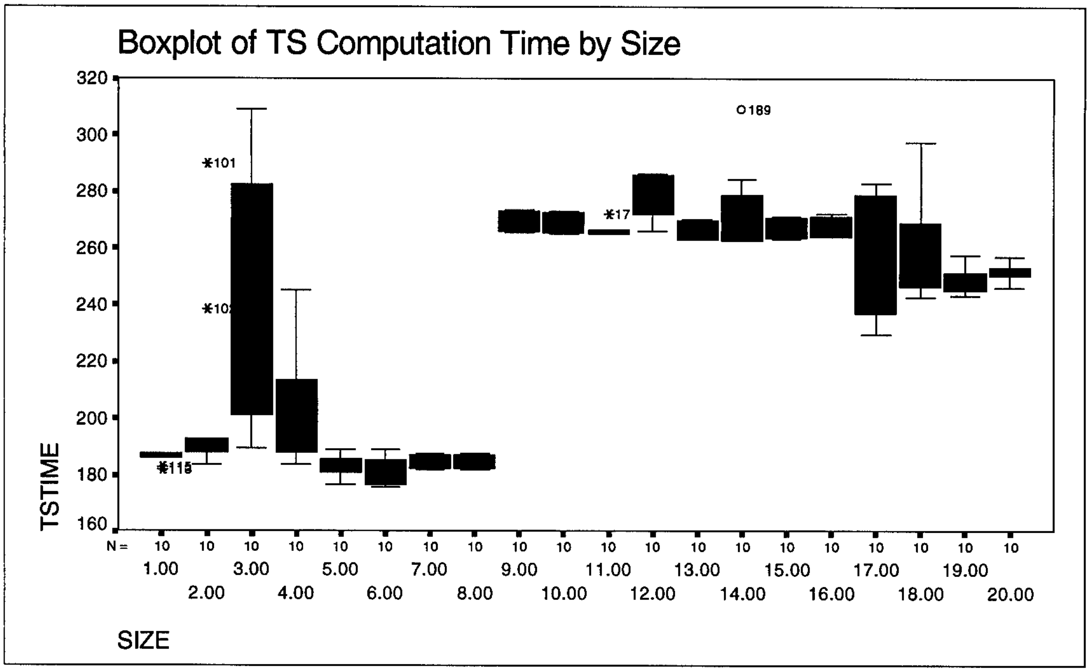

Figure 22. Boxplot of TS Computation Time by Size (CFLP)

"1" and the larger problems with a "2" and computed Spearman's rank correlation coefficients. The results are as expected; larger problems require longer computation time. Tabu search had a coefficient of .7626, simulated annealing had a coefficient of .6787, and the genetic algorithm had a coefficient of .8486. All were significant with a p = .000. In Figures 22, 23, and 24 we show boxplots generated by the SPSS software, which provide us with more information.

In each of the figures above, the number at the bottom identifies each of the problems. Problems 1 through 8 are the smaller 16x50 problems, and problems 9 through 20 are the larger 25x50 problems. The most obvious observation is that the change in size causes a step change in the computation time, especially for tabu search and genetic algorithms. The step change is less pronounced for simulated annealing. The degree of variability for each problem is also of

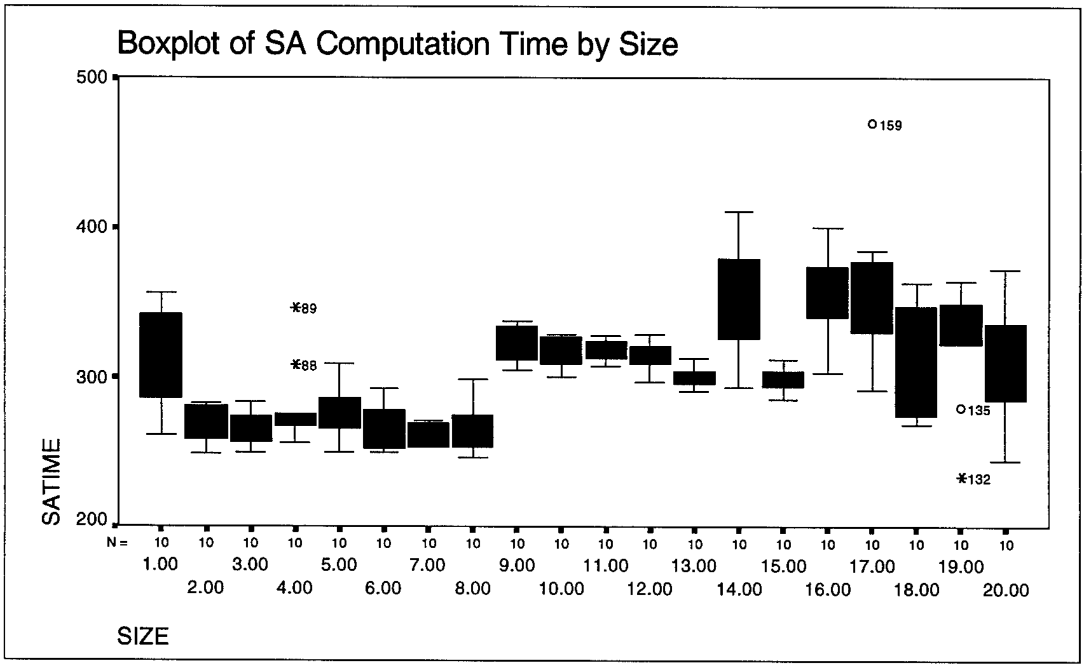

Figure 23. Boxplot of SA Computation Time by Size (CFLP)

interest. Although the genetic algorithm is the slowest of the three heuristics, it has almost no variability in computation time. Simulated annealing, as expected, has the most variability due to the particular implementation (recall for each epoch, a number of solutions between ACCEPT and FACTOR*ACCEPT are evaluated).

5.12 Chapter Summary

In this chapter we presented the implementation of three general heuristics for the capacitated facilities location problem. We then proceeded to conduct experiments to select an appropriate solution representation. Our experiments indicated the matrix/tableau representation was most appropriate when considering solution quality and computation time. Using this representation, we conducted a set of parameter setting experiments and reported on the various

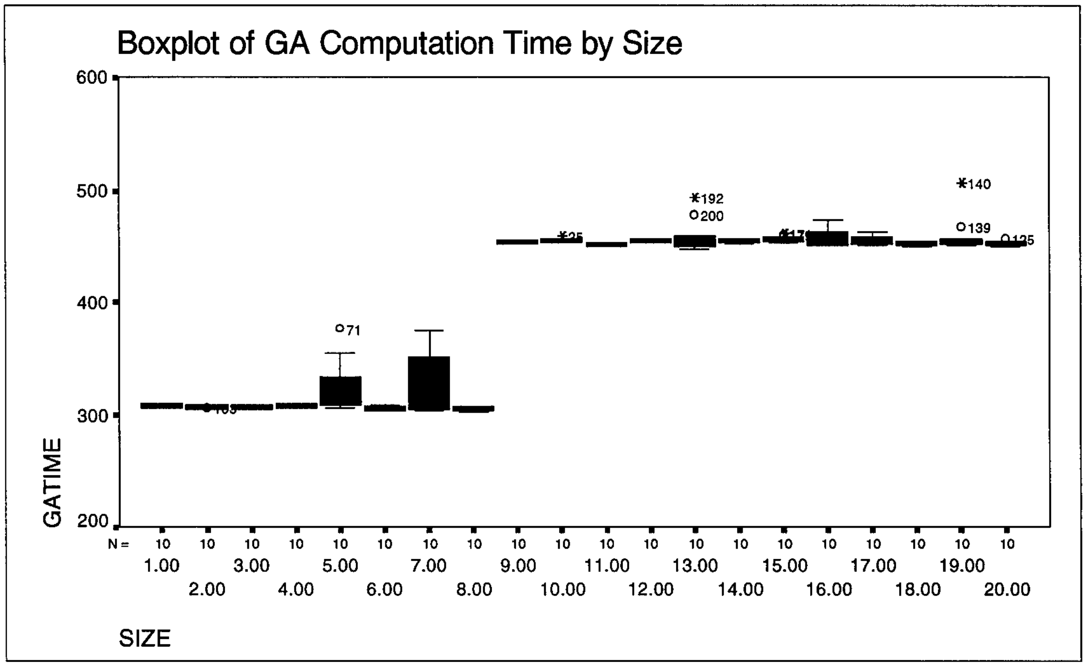

Figure 24. Boxplot of GA Computation time vs. Size (CFLP)

effects of parameter levels on solution quality and computation time. In most cases, parameters showed a significant impact on both of these measures. A subsequent set of benchmark experiments showed that tabu search and simulated annealing provide solutions close to optimal, with average optimality gaps of 2.32% and 3.17%. The genetic algorithm did not perform well, with an average optimality gap of 21.28%. We concluded the chapter with some observations about the effects of computation time on solution quality and problem size on computation time. The results of statistical tests are reported throughout.

In the next two chapters, we will address the multiple-period and multiple-commodities facilities location problems. In Chapter 8, we will report the results of using all heuristics on larger sized problems and determine the relative performance of each heuristic.

Chapter 6

HEURISTICS FOR MULTIPLE-PERIOD FACILITIES LOCATION PROBLEMS

In the previous chapter, we presented general heuristics for the capacitated facilities location problem (CFLP). This chapter parallels the previous chapter in presenting tabu search, simulated annealing, and genetic algorithm heuristics for the multiple period facilities location problem (MP-FLP). In the first section we introduce the MP-FLP, present a formulation, and mention some of the relevant literature. In the following sections, we present the heuristic algorithms. For each one we specify our particular implementation and parameters for each representation. Since much of this work is similar to the work done for CFLP, we make references to sections in Chapter 5 whenever it is appropriate. After all three heuristics have been explained, we report the results from the first three stages of experimentation: selection of appropriate representation, parameter experimentation, and benchmarking.

6.1 The Multiple-Period Facilities Location Problem

For MP-FLP, we wish to determine which of N capacity constrained locations to operate so we can satisfy the demands of M customers over a horizon of V periods at the lowest sum of fixed, variable, opening, and closing costs. The literature on this topic is rather limited. Warszawski (1973) presented optimal and heuristic methods for solving an uncapacitated version of the problem.

Roodman and Schwarz (1975, 1977) developed branch and bound algorithms for a version where existing facilities are to be phased out over a time horizon and also for a version where existing facilities may be phased out while other potential locations may be phased in over a time horizon. Neither of these versions involved capacity constraints. Our work is based primarily on a formulation by Sweeney and Tatham (1976) which allows opening or closing facilities at a potential location in any period with a corresponding cost. Their formulation is for a problem where intermediate warehouse locations must be selected to satisfy customer demands when there is also an existing set of factories which supply the warehouses. Their solution involves generating several static warehouse location problem solutions for each period, and using dynamic programming to select the best configuration at each period that will result in the overall lowest cost. Hormozi and Khumawala (1996) improve on the work of Sweeney and Tatham by reducing the number of static solutions that must be considered by the dynamic programming portion of the algorithm. Our formulation of MP-FLP is based on these two papers. For a problem with V periods, N potential locations, and M customers, the formulation is:

$$\begin{aligned}\min\ Z = &\sum_{t=1}^{V}\sum_{i=1}^{N}\sum_{j=1}^{M} C_{tij}x_{tij} + \sum_{t=1}^{V}\sum_{i=1}^{N} F_i y_{ti} + \sum_{i=1}^{N} OC_i y_{1i} \\ &+ \sum_{t=2}^{V}\sum_{i=1}^{N} OC_i y_{ti}\left(1 - y_{(t-1)i}\right) + \sum_{t=1}^{V-1}\sum_{i=1}^{N} CC_i y_{ti}(1 - y_{(t+1)i})\end{aligned} \tag{13}$$

subject to

$$\sum_{i=1}^{N} x_{tij} = D_{tj} \qquad j = 1,2,\ldots,M; \quad t = 1,2,\ldots,V \tag{14}$$

$$\sum_{j=1}^{M} x_{tij} \le S_i y_{ti} \qquad i = 1,2,\ldots,N; t = 1,2,\ldots,V \tag{15}$$

$$y_{ti} \in \{0,1\} \qquad i = 1,2,\ldots,N; \quad t = 1,2,\ldots,V \tag{16}$$

$$x_{tij} \geq 0 \qquad i = 1,2,\ldots N; \quad j = 1,2,\ldots,M; \quad t = 1,2,\ldots,V \tag{17}$$

where

D_{tj}	customer j's demand in period t
S_i	location i's capacity
C_{tij}	per unit cost to satisfy customer j's demand from location i
F_i	fixed cost of operating a warehouse at location i
OC_i	cost to open location i
CC_i	cost to close location i
x_{tij}	demand for customer j shipped from location i in period t
y_{ti}	$\{0,1\}$ indicator where $y_{ti}=1$ implies a warehouse is located at location I during period t

Specifically, fixed costs (F_i), variable costs (C_{tij}), opening costs (OC_i), closing costs (CC_i), location capacities (S_i), and customer demands (D_{tj}) are the given input data for the problem and the demand allocation (x_{tij}) and facility locations (y_{tij}) are the decision variables. Notice that in our formulation we do not vary capacity, fixed costs, open costs, nor closing costs over time periods.

6.2 Solution Representations

As we previously discussed, tabu search, simulated annealing, and genetic algorithms require an abstract representation of the problem's solution in order to define "moves" and "operators." In the following subsections we introduce two possible representations for MP-FLP: a 0-1 vector and a matrix.

6.2.1 0-1 Vector Representation

The 0-1 vector representation is directly related to the y_{ti}'s in the problem formulation. A particular solution is represented by a vector $\mathbf{s} = \{y_{11}, y_{12},\ldots,y_{ti}\}$, where $y_{ti} \in \{0,1\}$ indicating if a particular location is in operation on a particular period. Once this vector is determined, the x_{tij}'s can be determined by solving the transportation problem indicated by the set of open locations for each period.

With this information, the total costs for this solution can be computed using (13) above.

6.2.2 A Matrix Representation

A matrix representation has a direct connection to the x_{tij}'s in the problem formulation. Each row in the matrix represents a location, and each column one of the customers. The first M columns represent the first period, the second M columns represent the second period, and so on. The matrix is filled by assigning values to each element such that the sum of each column matches the corresponding customer's demand, and the sum of each row (by groups of M columns) does not exceed the corresponding location's capacity. A matrix thus filled represents a solution. The y_{ti}'s can be easily determined by examining the values in the matrix to determine which locations have been assigned some demand in any period. With this information, the total costs can be computed using (13) above.

6.3 A Tabu Search Heuristic for MP-FLP

Previously, we defined a tabu search heuristic as the following set of elements:

$$\text{TS} = \{ \mathscr{R}, \mathscr{N}(), \mathscr{C}(), \mathscr{A}(), \mathscr{T}(), \mathscr{P} \}$$

Where

$\mathscr{R}$	is the solution representation structure
$\mathscr{N}()$	is the neighborhood structure;
$\mathscr{C}()$	is the cost function
$\mathscr{A}()$	is the aspiration criterion function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

In the previous section we discussed two representations and their associated cost functions. The aspiration critera, terminating condition function,

and control parameters are the same as for CFLP which are described in §5.3.2, §5.3.3, and §5.3.4. It only remains to describe the neighborhood structure.

For a 0-1 vector solution, a neighbor is defined as a solution where the value of a single element in the vector has been changed from zero to one, or vice-versa. Thus, for a problem with n potential locations and v periods, a solution has a neighborhood size of nv. In our implementation, the entire neighborhood is evaluated at each iteration of the algorithm. Neighbors that do not represent a feasible solution (i.e., too few locations open in a period to meet customers' demands) are ignored.

For a matrix solution representation, a neighbor is defined as a change to two of the matrix elements within a column while maintaining the demand/capacity constraints. This change amounts to a reallocation of demand from one facility to another for a single customer in a single period. This is a very large and variable neighborhood since there may be as many reallocations possible as there are positive elements in the matrix, and each reallocation can be by as little as a quantity of one or as much as the value of the source element. Accordingly, we only evaluate a subset of this neighborhood at each iteration. The subset consists of making one reallocation for each column in the matrix for a neighborhood size of nv. The two elements involved in the reallocation are randomly selected, and the quantity of demand reallocated is the minimum of the source element's value or the destination element's remaining capacity. This procedure always results in feasible solutions.

6.4 A Simulated Annealing Heuristic for MP-FLP

In Chapter 2, we defined a simulated annealing algorithm as the following set of elements:

SA = { $\mathcal{R}, \mathcal{N}(), \mathcal{C}(), \mathcal{S}(), \mathcal{T}(), \mathcal{P}$ }

where

$\mathcal{R}$	is the solution representation structure
$\mathcal{N}()$	is the neighborhood structure;
$\mathcal{C}()$	is the cost function
$\mathcal{S}()$	is the cooling schedule
$\mathcal{T}()$	is the terminating condition function
$\mathcal{P}$	is the set of control parameters

Solution representations and cost functions were described in §6.2. The cooling schedule, terminating condition function, and control parameters are the same as for CFLP and are described in §5.4.2, §5.4.3, and §5.4.4. We now describe the neighborhood structure.

The neighborhood structure for simulated annealing is the same as for tabu search, except that we only evaluate and make a decision based only on a single, randomly selected neighbor at a time. For the 0-1 vector representation, we randomly select one element in the vector and change its status. If the resulting neighbor represents a feasible solution, we continue to evaluate it; otherwise, we select another one. For the matrix representation, we randomly select a column in the matrix and randomly select a positive element in this column. We then randomly select a destination element in the same column, and reallocate from the source to the destination the minimum of the source element's value or the remaining capacity in the destination element. This procedure always results in a feasible neighbor which is further evaluated.

6.5 A Genetic Algorithm Heuristic for MP-FLP

In Chapter 2, we defined a genetic algorithm as the following set of elements:

GA = { $\mathscr{R}$, $\mathscr{F}()$, $\mathscr{X}()$, $\mathscr{M}()$, $\mathscr{T}()$, $\mathscr{P}$}

Where

$\mathscr{R}$ is a solution representation
$\mathscr{F}()$ is the fitness function
$\mathscr{X}()$ is the crossover operation function
$\mathscr{M}()$ is the mutation operation function
$\mathscr{T}()$ is the terminating condition function
$\mathscr{P}$ is the set of control parameters

Solution representations are discussed in §6.2. The fitness function, terminating condition, parameters are the same as for CFLP and are discussed in §5.5.1, §5.5.4, and §5.5.5. We now describe the crossover and mutation operators.

6.5.1 Crossover Operators

The crossover operator is applied probabilistically to two parent chromosomes in the current generation to create two offspring for the new generation. With probability XOVER, the crossover operator is applied to the parents to generate the first offspring. With complementary probability, the first offspring is a copy of the first parent. With probability XOVER, the crossover operator is applied to the parents in reverse order to generate the second offspring. With complementary probability, the second offspring is a copy of the second parent. The crossover operation itself is a function of the solution representation.

For the 0-1 vector, we implement a single-point crossover operator of periods. Specifically, we randomly select $x \in \{1,\ldots,V\}$; the first offspring consists of all elements from the first parent which represent location status for periods

less than x, and all elements from the second parent which represent location status for periods x and above. For the second offspring, the roles of the parents are reversed. Both offspring are always feasible.

For the matrix representation, we also implement a single-point crossover. We select a crossover point x as above, and the first offspring consists of the matrix columns from the first parent representing customer demand allocations for periods less than x and the columns from the second parent representing customer demand allocations for periods x and above. For the second offspring, the roles of the parents are reversed. Again, the resulting offspring are always feasible.

6.5.2 Mutation Operators

The mutation operator is applied probabilistically to each gene of a chromosome, where a gene is defined as a period in the solution. This means a single chromosome may get as many as V mutations, but the probability of this event is small, $(\text{MUTATE})^V$. For the 0-1 vector representation, a period's mutation in a solution consists of randomly selecting a location and changing its operating status. In the case the selected location is currently "open," the mutation will only take place if closing the location will result in a feasible solution. Therefore, the effective rate of mutation is somewhat less than the MUTATE parameter specifies.

For the matrix representation, a period's mutation consists of randomly selecting a column (customer) for this period, randomly selecting source and destination elements, and making a demand reallocation equal to the minimum of

the source element's value or the destination element's remaining capacity. The mutation always results in a feasible solution.

6.6 Benchmark Test Problems

A set of 20 medium-sized test problems was created for the purposes of representation selection, parameter effects experimentation, and benchmarking. The data for the test problems was obtained from a database of 250 U.S. cities with the inter-city distance and 1990 Census population. For a NxM problem, the source locations (potential warehouse locations) are the top N largest cities in terms of population while the customer locations are the first M cities arranged alphabetically. In this way, source locations are not necessarily also customer destinations. The set of 20 benchmark problems is summarized in Table 21. We now describe each of the test problem attributes:

Size: the problem dimensions were selected to match those found in Hormozi and Khumawala (1996). Their problem set has problems with 10, 15, or 20 periods, 10, 11, or 12 locations, and 25, 28, or 30 customers (2.5 times the number of locations). From this mix, we selected five combinations: 10 periods x 10 locations x 25 customers, 10x12x30, 15x10x15, 15x12x30, and 20x10x25.

Demand: the first period's demand is a fixed fraction of the city's population. The remaining period's demand follow one of five different demand patterns, again based on the work of Hormozi and Khumawala: increasing, decreasing, concave, convex, and seasonal.

Capacity: for each problem, the capacity assigned to each location is the same for all locations. For each problem size, two levels of capacity constraints

Table 21. Benchmark Test Problems (MP-FLP)

Problem	Size	Capacity	Demand	Open	Close
MP201	10x10x25	2500	Increasing	4418	3313
MP202	10x10x25	2500	Increasing	2121	1590
MP203	10x10x25	5000	Increasing	4418	3313
MP204	10x10x25	5000	Increasing	2121	1590
MP205	10x12x30	2500	Decreasing	3692	2769
MP206	10x12x30	2500	Decreasing	1772	1329
MP207	10x12x30	5000	Decreasing	3692	2769
MP208	10x12x30	5000	Decreasing	1772	1329
MP209	15x10x25	2500	Concave	4804	3603
MP210	15x10x25	2500	Concave	2306	1729
MP211	15x10x25	5000	Concave	4804	3603
MP212	15x10x25	5000	Concave	2306	1729
MP213	15x12x30	2500	Convex	4645	3484
MP214	15x12x30	2500	Convex	2230	1672
MP215	15x12x30	5000	Convex	4645	3484
MP216	15x12x30	5000	Convex	2230	1672
MP217	20x10x25	3000	Seasonal	3716	2787
MP218	20x10x25	3000	Seasonal	1784	1338
MP219	20x10x25	6000	Seasonal	3716	2787
MP220	20x10x25	6000	Seasonal	1784	1338

are established. A tight level consists of assigning capacity to each location such that their total sum for each period is approximately twice the total demand from all customers for that period. A loose level consists of twice the capacity allowed for the tight level.

Fixed Costs: for each problem size two levels of fixed costs are assigned. The basic level is based on the work of Hormozi and Khumawala and consists of a fixed costs based on the average throughput cost. The second level is half the basic level.

Opening/Closing Costs: these costs are also set according to Hormozi and Khumawala. The opening cost for all locations is 50% of the average fixed cost of all locations. The closing cost is 75% of the open cost.

Variable Costs: these costs are set at 2.5 cents per mile, where the distance between any source and destination is obtained from the database. This is the

Table 22. Parameters for Representation Selection (MP-FLP)

	0-1 Vector	Matrix
TSMP		
LTM	5	5
STM	20	2000
TLSIZE	7	7
SAMP		
RATE	.8	.85
ACCEPT	20	1000
FACTOR	2	3
GAMP		
POP	30	40
GENS	75	1500
XOVER	.85	.80
MUTATE	.05	.05

same cost factor used in the CFLP problems based on Khuen and Hamburger (1963).

6.7 Representation Selection

We conducted an experiment in order to determine which of the two problem representations would be most appropriate for MP-FLP. In this experiment, each of the three general heuristics was used to solve each of the 20 benchmark problems described above using each of the problem representations, resulting in an experiment with 120 cells. Due to the number of cells, we only completed five replications per cell. The parameters used for each of the six heuristic-representation combinations are shown in Table 22.

In order to determine which representation would be most appropriate, we first ranked the performance of each representation on each problem by heuristics (with respect to solution quality and computation time). The results of these rankings are shown in Table 23. The raw results from which these

Table 23. Representation Performance Ranks (MP-FLP)

TS Solution Ranks		SA Solution Ranks		GA Solution Ranks		TS Time Ranks		SA Time Ranks		GA Time Ranks	
V	T	V	T	V	T	V	T	V	T	V	T
2	1	2	1	2	1	2	1	2	1	1	2
1	2	1	2	2	1	2	1	2	1	1	2
2	1	2	1	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	1	2	2	1	2	1	2	1
2	1	1	2	1	2	2	1	2	1	2	1
2	1	1	2	1	2	2	1	2	1	1	2
2	1	2	1	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	2	1
2	1	1	2	2	1	2	1	2	1	2	1
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2
2	1	1	2	1	2	2	1	2	1	1	2
2	1	1	2	1	2	2	1	2	1	1	2
2	1	1	2	1	2	2	1	2	1	1	2
2	1	1	2	2	1	2	1	2	1	1	2

rankings come are shown in Tables 98, 99, and 100 at Appendix A. The rankings show the matrix representation (T) performs best for the tabu search and genetic algorithm heuristics, while the 0-1 vector (V) representation performs best for simulated annealing. The time performance ranks clearly indicate the matrix representation is preferred for tabu search and simulated annealing.

We computed Friedman's F_r test statistic for a randomized block design for the solution ranks without regard for heuristic. This results in an experiment with 60 blocks and 2 treatments. The null hypothesis states there is no difference between the treatments. The F_r statistic is 1447.2 which is larger than χ^2 = 3.84146 (p=.05, 1df) and we thus reject the null hypothesis. Since the matrix

representation has the smaller sum of ranks and its time performance is preferred, we select the matrix representation as most appropriate for MP-FLP.

6.8 Parameter Effects

As we did for CFLP in Chapter 5, we conducted an experiment to get a clear picture of how various parameter settings affect solution quality and computation time. We also use the results from these experiments to select the parameter settings that we use for benchmarking and comparative runs.

For tabu search, we experimented with the same parameter levels as we did for CFLP. These were three levels for long-term memory (LTM), five levels for short-term memory (STM), and three levels of maximum tabu list size (TLSIZE). The experimental levels are summarized in Table 24.

Table 24. TS Parameters Experimental Levels (MP-FLP)

Parameter	Experimental Levels				
LTM	1	5	10		
STM	1000	1500	2000	2500	3000
TLSIZE	5	7	12		

For simulated annealing, we also experimented with the same levels as for CFLP. These were five levels for the RATE parameter, four levels for the ACCEPT parameter, and three levels for the FACTOR parameter. The experimental levels are summarized in Table 25.

Table 25. SA Parameters Experimental Levels (MP-FLP)

Parameter	Experimental Levels				
RATE	0.75	0.80	0.85	0.90	0.95
ACCEPT	1000	1500	2000	2500	
FACTOR	1	2	3		

For genetic algorithms, we experimented with three levels for the POP, GENS, and XOVER parameters, and two levels for the MUTATE parameter. The experimental levels are summarized in Table 26.

Table 26. GA Parameters Experimental Levels (MP-FLP)

Parameter	Experimental Levels		
POP	20	30	40
GENS	1000	1500	2000
XOVER	0.80	0.85	0.95
MUTATE	0.05	0.10	

To conduct these experiments, we selected one of the benchmark test problems as a test bed. Since the results from the parameter experiment would lead to the selection of parameters to be used in solving the set of larger problems, we selected one of the largest benchmark problems, MP219, which was also apparently the most difficult to solve of the four problems of this size.

The tabu search experiment has 45 cells, the simulated annealing experiment has 60 cells, and the genetic algorithm experiment has 54 cells. Considering the number of experimental cells, we limited each cell to 10 replications.

To determine if there is a statistically significant difference among the experimental levels for each parameter with respect to solution quality and computation time, we computed the Kruskal-Wallis (K-W) statistic. To determine the general direction of the effect and its significance, we also computed Spearman's rank correlation coefficient for each parameter effects. The results of these statistical tests are summarized in Tables 27, 28, 29, and 30 and are

Table 27. Kruskal-Wallis H Test of Heuristic Parameters Effect on Solution Quality (MP-FLP)

Parameter	Chi-Square	D.F.	Significance
LTM	138.6224	2	.0000
STM	12.4791	4	.0141
TLSIZE	0.1422	2	.9314
RATE	233.9177	4	.0000
ACCEPT	80.8984	3	.0000
FACTOR	154.1628	2	.0000
POP	0.4648	2	.7926
GENS	420.0246	2	.0000
XOVER	0.4089	2	.8151
MUTATE	53.7330	1	.0000

Table 28. Kruskal-Wallis H Test of Heuristic Parameters Effect on Computation Time (MP-FLP)

Parameters	Chi-Square	D.F.	Significance
LTM	369.3040	2	.0000
STM	73.7951	4	.0000
TLSIZE	0.0145	2	.9928
RATE	353.7137	4	.0000
ACCEPT	115.5075	3	.0000
FACTOR	109.6259	2	.0000
POP	265.7146	2	.0000
GENS	243.7994	2	.0000
XOVER	0.2734	2	.8722
MUTATE	6.3019	1	.0121

discussed in the following sections. The raw results tables can be found in Appendix B. Boxplots of the effects can be found at Appendix D.

6.8.1 Tabu Search LTM Parameter

The effects of the LTM parameter on solution quality are evident from Figure 25. There is a significant improvement in solution quality going from one long-term memory pass to five passes. The improvement going from five to ten passes, however, is not as dramatic. The K-W statistic of 138.6224 ($p = .000$) indicates a significant difference and the Spearman's correlation coefficient of -.5427 ($p = .000$) confirms the trend.

Table 29. Spearman Correlation Coefficients of Heuristic Parameters vs Solution Quality

Parameter	Coefficient	Significance
LTM	-.5427	.000
STM	-.1500	.001
TLSIZE	-.0047	.921
RATE	-.6114	.000
ACCEPT	-.3668	.000
FACTOR	-.5021	.000
POP	.0102	.814
GENS	-.8754	.000
XOVER	-.0275	.524
MUTATE	.3157	.000

Table 30. Spearman Correlation Coefficient of Heuristic Parameters vs Computation Time

Parameter	Coefficient	Significance
LTM	.9045	.000
STM	.4039	.000
TLSIZE	.0052	.912
RATE	.7595	.000
ACCEPT	.4363	.000
FACTOR	.4208	.000
POP	.6987	.000
GENS	.6632	.000
XOVER	.0112	.796
MUTATE	.1081	.012

Figure 26 shows the impact of LTM on computation time. As expected, more long-term memory passes take significantly more time, especially as the STM parameter itself is increased.

6.8.2 Tabu Search STM Parameter

As the STM parameter is increased, one would expect each line in Figure 25 to slope downwards indicating an improvement in solution quality due to the higher number of solutions evaluated. Although such a downward slope is present, it is not a very strong effect as evidenced by a correlation coefficient of -.1500 ($p = .001$). The K-W test, however, does indicate a significant effect with a statistic of 12.4791 ($p = .0141$).

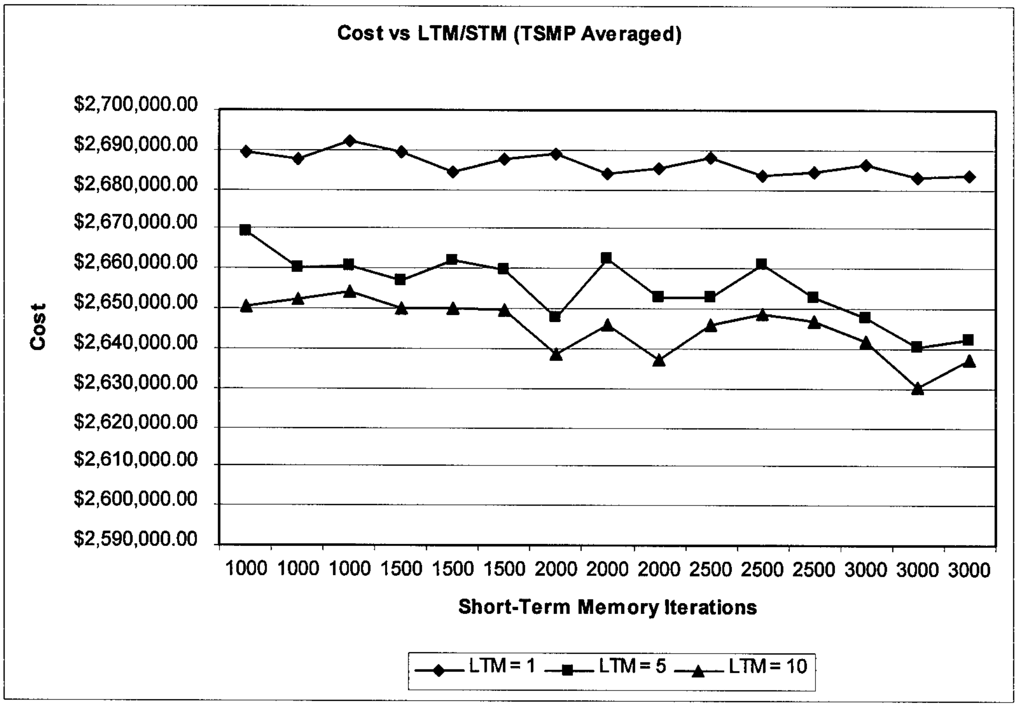

Figure 25. TS Parameters Effect on Cost (MP-FLP)

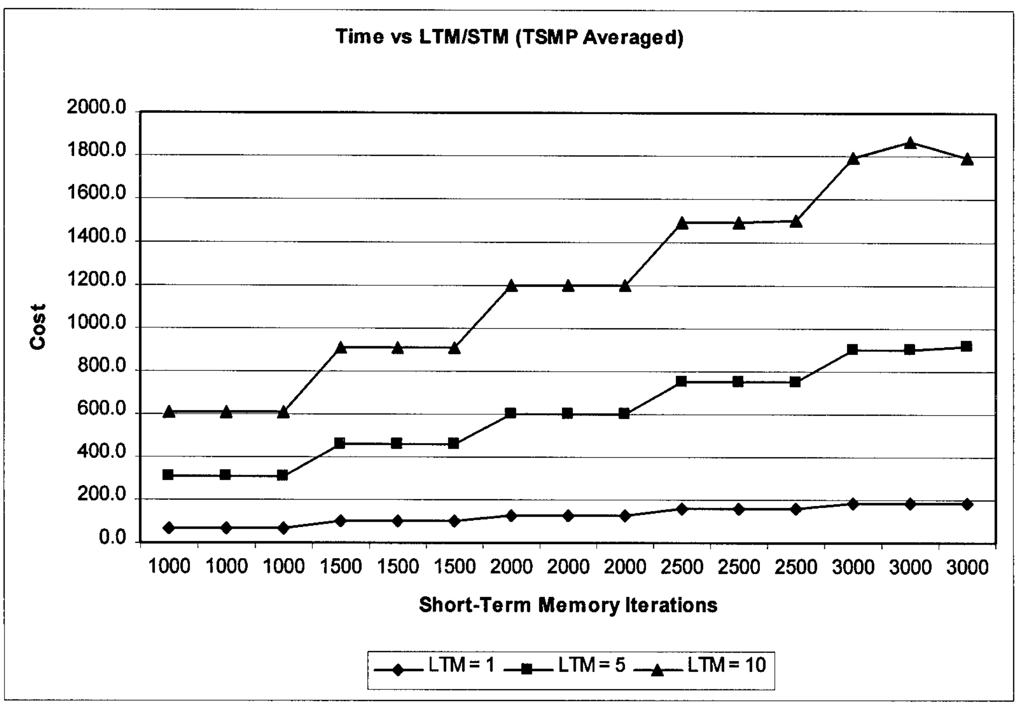

Figure 26. TS Parameters Effect on Time (MP-FLP)

Figure 26 clearly shows the significant effect the STM parameter has on computation time. One would expect to see an upward sloping line indicating longer computation time due to the increase in the number of solutions evaluated. Although the effect is not very strong for the case when LTM = 1, it is very strong for the other two levels of the LTM parameter since it has a multiplicative effect. The correlation coefficient of .4039 (p = .000) and K-W statistic of 73.7951 (p = .0000) confirm these observations.

6.8.3 Tabu Search TLSIZE Parameter

Figure 25 indicates that there is not much of a consistent pattern to the effect of TLSIZE on solution quality. For each line representing a level of the LTM parameter, the effects of the TLSIZE parameter can be seen by examining the individual points in groups of three corresponding to a single level of the STM parameter an indicated on the x-axis. For the case where LTM = 1, there seems to be a consistent improvement for the middle point corresponding to TLSIZE = 7. However, for the other two levels of LTM, the pattern changes as the levels of the STM parameter change. There is not a significant correlation between this parameter and solution quality as evidenced by the correlation statistic of -.0047 (p = .921). Similarly, the K-W test finds no significant difference between the levels of TLSIZE, with a statistic of .1422 (p = .9314).

TLSIZE does not have a significant effect on computation as can be seen in Figure 26. Each of the three-point groupings for the three levels of TLSIZE consists of a flat line indicating no impact on computation time. Accordingly, the

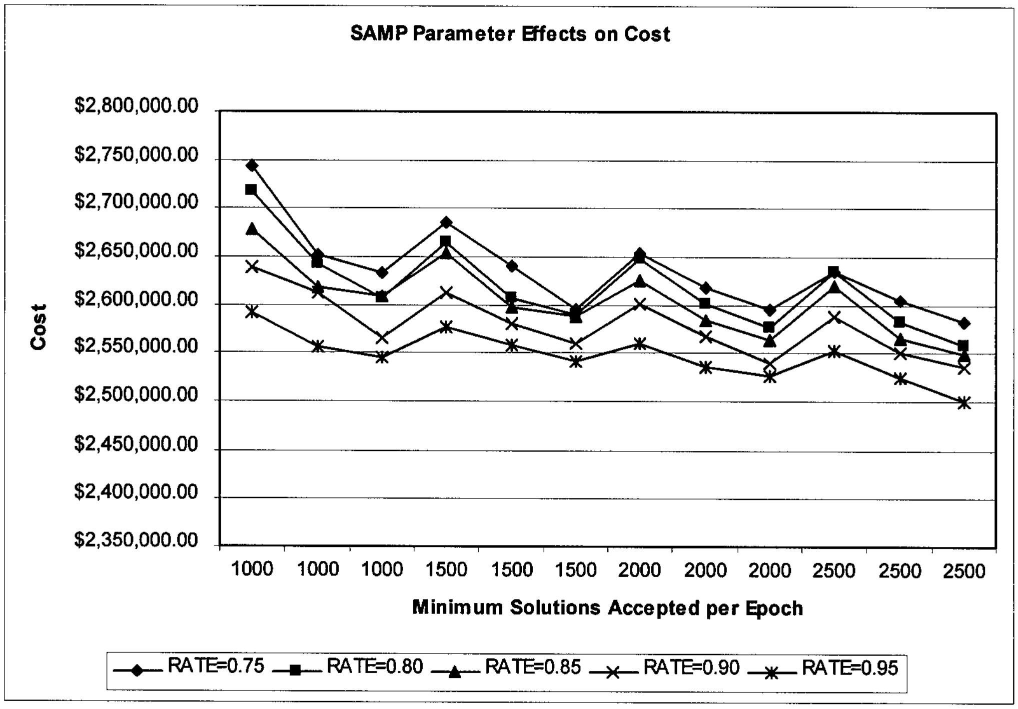

Figure 27. SA Parameters Effect on Cost (MP-FLP)

correlation coefficient is .0052 (p = .912) and the K-W statistic is .0145 (p = .9928).

6.8.4 Tabu Search Selected Parameters

Based on the preceding results, we selected the following parameter settings for all future experiments: LTM = 5, STM = 3000, and TLSIZE = 7. Although a setting of LTM = 10 gives slightly better results (0.4% savings), the extra computation time is not justified (107% increase).

6.8.5 Simulated Annealing RATE Parameter

The effects of the RATE parameter are shown in Figure 27. Each line represents a level of RATE. Although there seems to be some overlap for RATE = 0.80 and RATE = 0.85, it is clear that as the RATE parameter increases, solution quality improves. In particular, the best results are achieved by RATE =

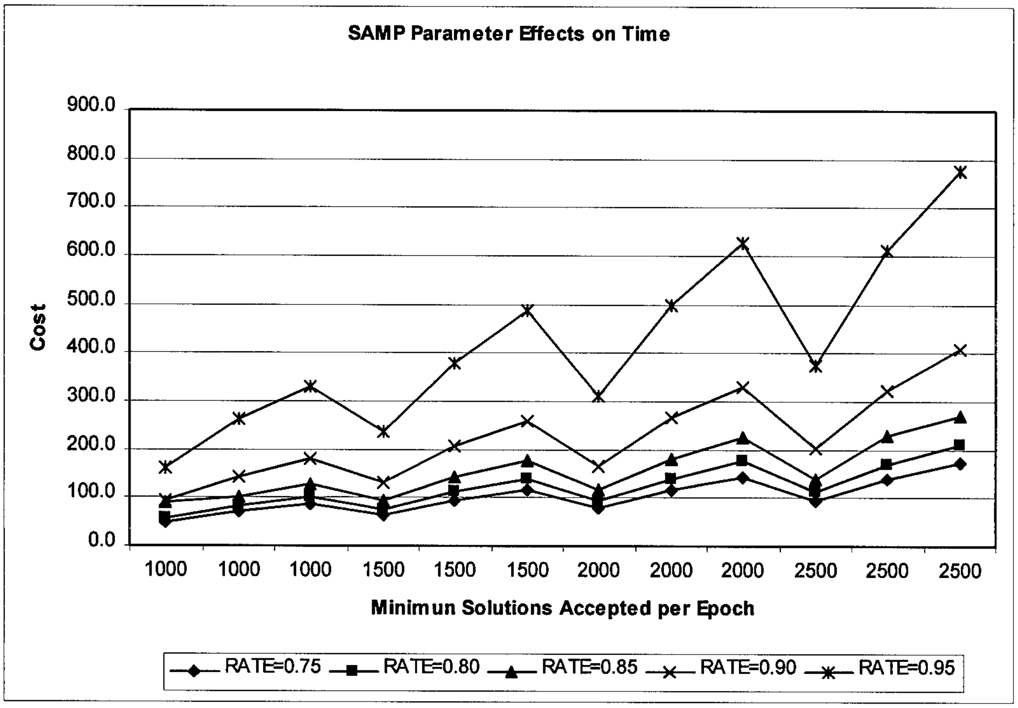

Figure 28. SA Parameters Effect on Time (MP-FLP)

0.95. The correlation coefficient is -.6114 (p = .000) and the K-W statistic is 233.9177 (p = .0000).

The effects of RATE on computation time are shown in Figure 28. As expected, as the RATE parameter is increased, more time is required to complete each run of the algorithm due to the increase in solutions evaluated resulting from a slower cooling schedule. Note that the effect is most dramatic for RATE = 0.95 and less so for the other levels. The correlation coefficient is .7595 (p = .000) and the K-W statistic is 353.7137 (p = .0000).

6.8.6 Simulated Annealing ACCEPT Parameter

The effect of the ACCEPT parameter on solution cost is represented by the slope of the lines in Figure 27. As expected, as the ACCEPT parameter is increased requiring more solutions to be evaluated, the solution quality improves.

This effect is confirmed by a Spearman's correlation coefficient of -.3668 (p = .000) and a K-W statistic of 80.8984 (p = .0000).

The effects of the ACCEPT parameter on computation time are represented in Figure 28 by the slope of each line. As expected, there is an upward slope indicating the need for more time as the ACCEPT parameter is increased. Again, this is confirmed by the correlation coefficient of .4363 (p = .000) and the K-W statistic of 115.5075 (p = .0000).

6.8.7 Simulated Annealing FACTOR Parameter

The effects of the FACTOR parameter on solution quality are shown in Figure 27. Looking at the three points corresponding to each grouping of the ACCEPT parameter on the x-axis, it is clear that as the FACTOR parameter is increased, solution quality improves. This pattern is consistent for all levels of RATE and ACCEPT parameters. The correlation coefficient is -.5021 (p = .000) and the K-W statistic is 154.1628 (p = .000)

The effects of the FACTOR parameter on computation time are also clearly evident in Figure 28. Looking at the same three-point groupings, each group shows a clear upward slope indicating the need for more computation time as the FACTOR parameter increases allowing more solutions to be evaluated. The correlation coefficient of .4208 (p = .000) and K-W statistic of 109.6259 (p = .0000) confirm these observations.

6.8.8 Simulated Annealing Selected Parameters

Based on the preceding results, we selected the following parameter settings for future experiments: RATE = 0.90, ACCEPT = 2500, and FACTOR = 3. These parameter settings give good results within reasonable computation times.

6.8.9 Genetic Algorithm POP Parameter

Looking at Figure 29, it would appear that the POP parameter does not have a great impact on solution quality, especially when the number of generations is greater than 1000. Each line represents a combination of the POP and MUTATE parameters (POP = 20A indicates a population of 20 and a MUTATE parameter of 0.05). The lines are split into two groups according to the MUTATE parameter setting. For each level of the MUTATE parameter, however, the lines are tightly grouped showing a lack of impact for the level of POP. Accordingly, neither the correlation coefficient of .0102 ($p = .814$) nor the K-W statistic of 0.4648 ($p = .7926$) show a statistically significant impact or difference.

With respect to computation time, Figure 30 clearly shows the impact of increasing the POP parameter. Each two-line grouping in Figure 30 indicates a level of the POP parameter. The lowest two lines are for POP = 20 and the top two lines are for POP = 40. Clearly, a larger population requires longer computation time. The correlation coefficient is .6987 ($p = .000$) and the K-W statistic is 265.7146 ($p = .0000$).

6.8.10 Genetic Algorithm GENS Parameter

The effect of increasing the GENS parameter is clearly shown in Figure 29. The downward sloping lines indicate the improvement on solution quality as more

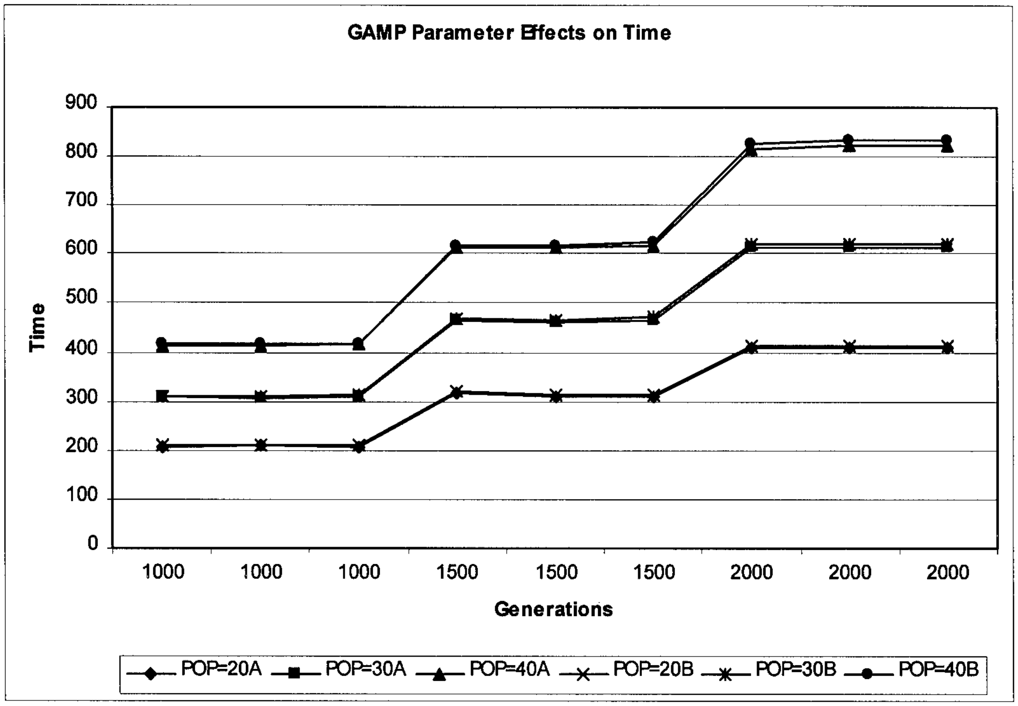

Figure 29. GA Parameters Effect on Time (MP-FLP)

generations are allowed. This is an expected result confirmed by a correlation coefficeint of -.8754 (p = .000) and a K-W statistic of 420.0246 (p = .0000).

The effect of GENS on computation time is also clearly visible in Figure 30. As more generations are allowed, more time is required to solve the increased number of solutions. A correlation coefficient of .6632 (p = .000) and a K-W statistic of 243.7994 (p = .0000) confirm these observations.

6.8.11 Genetic Algorithm XOVER Parameter

The effects of the XOVER parameter on solution quality can be seen in Figure 29. Each line consists of nine points, where each group of three points is a level of the GENS parameter, and each point in these groupings is a level of the XOVER parameter. The flat lines indicate that the XOVER parameter does not have a significant impact on solution quality. This is confirmed by the

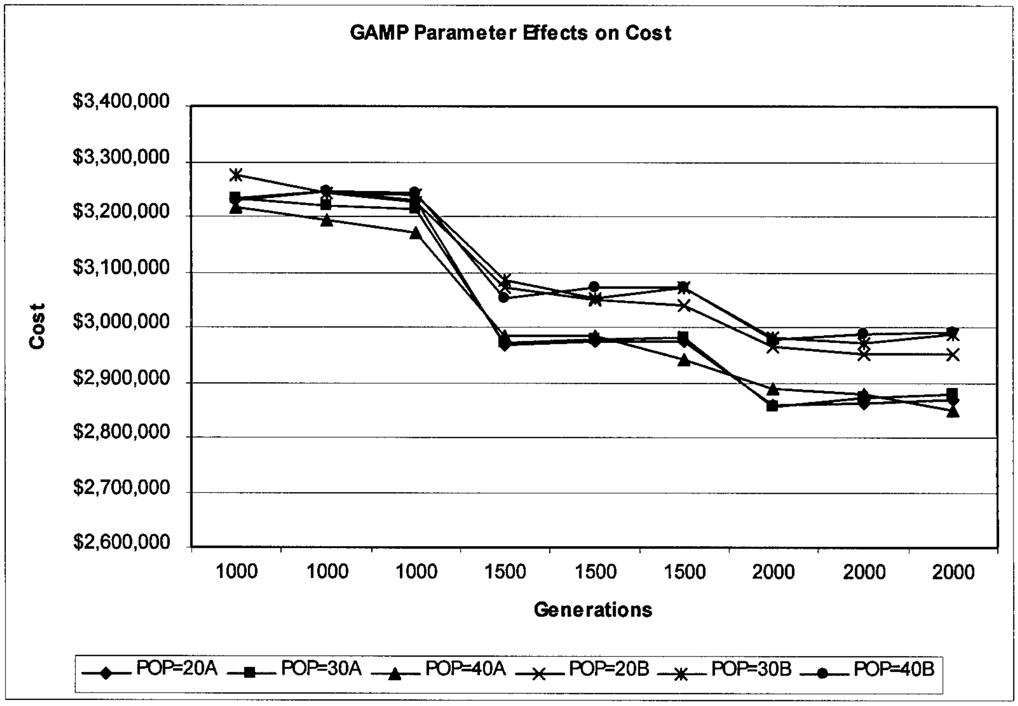

Figure 30. GA Parameters Effect on Cost (MP-FLP)

correlation coefficient of -.0275 (p = .524) and a K-W statistic of .4089 (p = .8151).

Looking at the same point groupings in Figure 30, it can be seen that the XOVER parameter also does not have a significant impact on computation time. A correlation coefficient of .0112 (p = .796) and K-W statistic of .2734 (p = .8722) confirm the lack of significant impact.

6.8.12 Genetic Algorithm MUTATE Parameter

The effects of the MUTATE parameter on solution quality are shown in Figure 29. Although the effect is not quite clear for the GENS level of 1000, it is very clear for the other two levels. The top group of lines, identified on the legend by a "B," correspond to the MUTATE = .1 rate. The lower group of lines, identified on the legend by an "A," correspond to the MUTATE = .05 rate.

Clearly, the .05 mutation rate results in superior performance. This is confirmed by a correlation coefficient of .3157 (p = .000) and a K-W statistic of 53.7330 (p = .0000).

With respect to time, Figure 30 indicates that the MUTATE parameter does not have a strong impact on computation time. Each pair of lines in Figure 30 corresponds to a level of the POP parameter. For each of these pairs, each line corresponds to a level of the MUTATE parameter. Visually, each pair of lines is almost a single line. Surprisingly, the K-W statistic of 6.3019 (p = .0121) shows there is in fact a significant difference between the two levels. The correlation coefficient of .1081 (p = .012) shows the larger mutation rate requires slightly more computation time.

6.8.13 Genetic Algorithm Selected Parameters

Based on the preceding results, we selected the following parameter settings for all future experiments: POP = 30, GENS = 2000, XOVER = 0.80, and MUTATE = 0.05. Although a POP = 40 setting gives a light improvement (0.2%), a 34% increase in computation time is not justified.

6.9 Benchmark Test Results

In this section we report the results from benchmarking experiments. We used the 20 benchmark test problems described earlier in Table 21. Each of these problems was first solved to optimality using an algorithm based on the work of Sweeney and Tatham (1976) and Hormozi and Khumawala (1996). We then solved each problem using each of the general heuristics with the matrix representation. Parameter settings were selected based on the results from the

previous parameter experiments and are summarized in Table 31. For each problem we completed ten replications using each heuristic. The average results and resulting gaps from optimality are reported in Table 32.

Table 31. Parameter Settings for Benchmark Experiments (MP-FLP)

TS Parameters		SA Parameters		GA Parameters	
LTM	5	RATE	0.90	POP	30
STM	3000	ACCEPT	2500	GENS	2000
TLSIZE	7	FACTOR	3	XOVER	0.80
				MUTATE	0.05

The results of the benchmark experiments show that tabu search has an average gap from optimality of 6.85%, simulated annealing has an average gap from optimality of 3.36%, and the genetic algorithm has an average gap from optimality of 15.23%. Clearly, the genetic algorithm did not perform well in most of the test cases. If we look at the time it took each algorithm to solve a problem, we see that both tabu search and genetic algorithm take longer as the size of the problem increases, whereas the time for simulated annealing remains rather constant. Also, all of the heuristics do best for problems with high capacity and low fixed costs (MP204, MP208, MP212, MP216, and MP220).

6.10 Computation Time vs. Solution Quality

To determine what kind of relationship exists between computation time and solution quality we again use the data collected during the parameter selection experiments. Tables 28 and 30 indicate that most parameters have a significant impact on computation time. The immediate by-product of these experiments is a set of data points with a range of computation times for the same problem with

Table 32. MP-FLP Benchmark Results

	Tabu Search			Simulated Annealing			Genetic Algorithm		
Problem	Solution	Opt Gap	Time	Solution	Opt Gap	Time	Solution	Opt Gap	Time
MP201	$1,508,492.40	10.57%	325	$1,430,813.70	4.87%	327	$1,638,614.60	20.11%	308
MP202	$1,206,264.50	7.56%	329	$1,141,657.80	1.80%	333	$1,286,019.50	14.67%	308
MP203	$1,341,270.40	6.04%	342	$1,283,184.00	1.45%	268	$1,425,114.20	12.67%	308
MP204	$1,070,359.80	1.92%	341	$1,057,780.40	0.72%	278	$1,112,377.50	5.92%	308
MP205	$1,196,251.30	10.99%	464	$1,149,933.40	6.69%	281	$1,295,901.50	20.24%	434
MP206	$ 885,670.80	1.95%	465	$ 881,667.50	1.49%	279	$ 937,243.20	7.88%	434
MP207	$1,185,676.80	11.12%	473	$1,122,445.10	5.19%	254	$1,252,594.00	17.39%	434
MP208	$ 883,612.10	2.23%	468	$ 874,819.10	1.22%	263	$ 920,307.10	6.48%	433
MP209	$2,542,588.60	10.83%	495	$2,448,582.90	6.74%	292	$2,815,695.90	22.74%	458
MP210	$2,007,475.00	5.90%	494	$1,943,085.00	2.51%	272	$2,214,988.40	16.85%	458
MP211	$2,303,861.90	6.93%	509	$2,218,006.70	2.94%	253	$2,466,739.60	14.49%	457
MP212	$1,827,691.70	2.34%	509	$1,815,193.90	1.64%	256	$1,922,802.80	7.66%	459
MP213	$2,755,949.40	12.62%	677	$2,614,394.80	6.84%	372	$3,090,503.70	26.30%	649
MP214	$2,103,823.50	10.48%	670	$2,006,435.80	5.37%	380	$2,367,430.30	24.32%	650
MP215	$2,387,461.00	9.49%	700	$2,294,093.50	5.20%	271	$2,630,672.40	20.64%	649
MP216	$1,818,745.20	2.44%	703	$1,800,462.30	1.41%	274	$1,940,847.90	9.32%	649
MP217	$2,714,527.40	8.29%	671	$2,591,823.60	3.40%	274	$2,989,322.90	19.25%	612
MP218	$2,190,317.30	4.06%	664	$2,151,288.80	2.20%	292	$2,351,930.70	11.74%	610
MP219	$2,640,209.90	7.95%	681	$2,535,707.10	3.68%	283	$2,856,778.20	16.81%	608
MP220	$2,141,757.90	3.20%	676	$2,113,810.70	1.85%	283	$2,264,084.00	9.09%	609

each heuristic. Thus, we can use Spearman's rank correlation coefficient to determine if there is a relationship between solution time and solution quality.

The correlation coefficients summarized in Table 33 below indicate that there is a significant negative correlation between computation time and solution quality for all heuristics.

Table 33. Spearman's Rank Correlation Coefficients for Computation Time vs. Solution Quality (MP-FLP)

Heuristic	Spearman's	Significance
Tabu Search	-.5625	.000
Simulated Annealing	-.8678	.000
Genetic Algorithm	-.5430	.000

In order to appreciate how these heuristics perform over time, we recorded the time at which each improvement is made as a heuristic solves the problem.

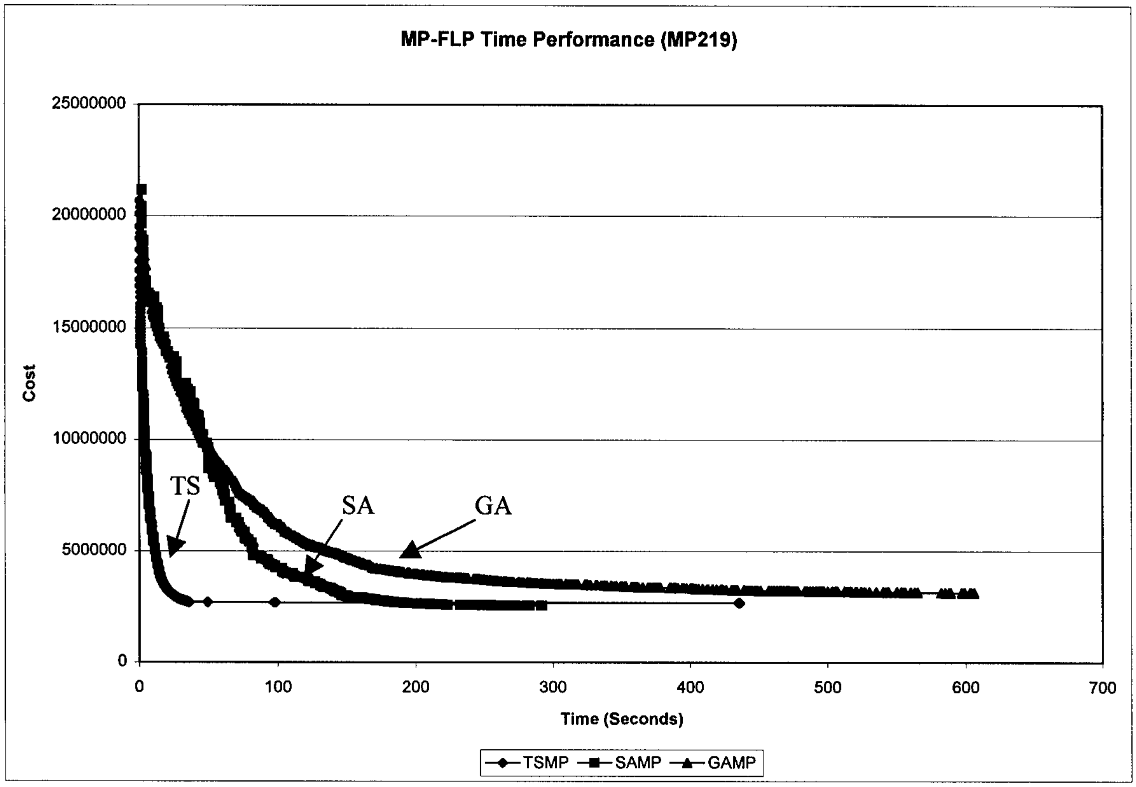

Figure 31. Heuristics Time Performance (MP-FLP)

In Figure 31 we show the time performance of each heuristic for benchmark problem MP219, the same problem used for the parameter experiments. The parameter settings for each heuristic are the same as for the benchmark experiments shown in Table 31.

Figure 31 shows that tabu search is the most efficient in finding good solutions with respect to time (i.e., it reaches good solutions rapidly). Simulated annealing and the genetic algorithm are more deliberate and slow in their search, although for this problem they eventually come close to the tabu search solution. Therefore, when computation time is limited, tabu search is clearly the preferred choice.

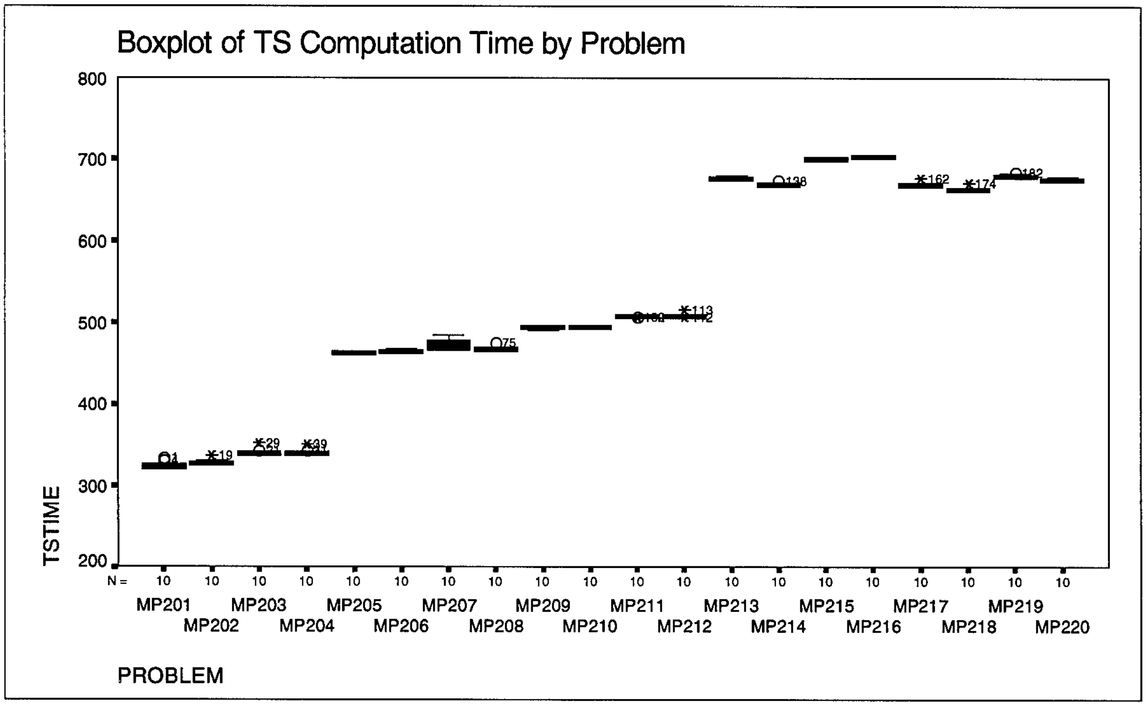

Figure 32. Boxplot of TS Computation Time by Problem (MP-FLP)

If we compare the heuristics' time performance for MP-FLP with that for CFLP (see Figure 21 in Chapter 5), we see that for MP-FLP the genetic algorithm is not as efficient and its performance resembles that of simulated annealing for CFLP. We will consider further the time performance of this heuristics in subsequent chapters as we evaluate the heuristics performance when time is limited.

6.11 Problem Size vs. Computation Time

To examine the relationship between problem size and computation time we use the data collected during the benchmarking experiments. These data was collected from four problems for each of five sizes as described in Table 21. We coded each problem size with a number from one through five according to the number of cells in the problem (the product of periods, locations, and customers).

Problems MP201-MP204 (2500 cells) were coded as 1; problems MP205-MP208 (3600 cells) were coded as 2; problems MP209-MP212 (3750 cells) were coded as 3; problems MP217-MP220 (5000 cells) were coded as 4; and, problems MP213-MP216 (5400 cells) were coded as 5. Note that the coding does not follow the problem numbering sequentially.

As expected, the results indicate that larger problems require more computation time. Tabu search and genetic algorithms had very high correlation coefficients of .9556 and .9799 respectively ($p = .000$ for both). Simulated annealing had a much lower coefficient of .1664 ($p = .019$). We can get a better picture of these effects by looking at the boxplots shown in Figures 32, 33, and 34. Looking at Figure 32, we can clearly see the effect of problem size on the computation time required by tabu search. We can also see that there is little variability in computation time as a result of using different random number streams.

In the case of simulated annealing (Figure 33), we get a much different picture. First we note that some problems have considerable computation time variability as a result of using different random number streams. Second, we note that we do not see a step change in computation times as we saw for tabu search. Third, there are four problems, MP201, MP202, MP213, and MP214, which require much more computation time than the other problems. The only factor these problems share in common is that they have a tight capacity constraint level. Finally, genetic algorithms show an effect nearly identical to that of tabu search, with noticeable step changes and very little variability (Figure 34).

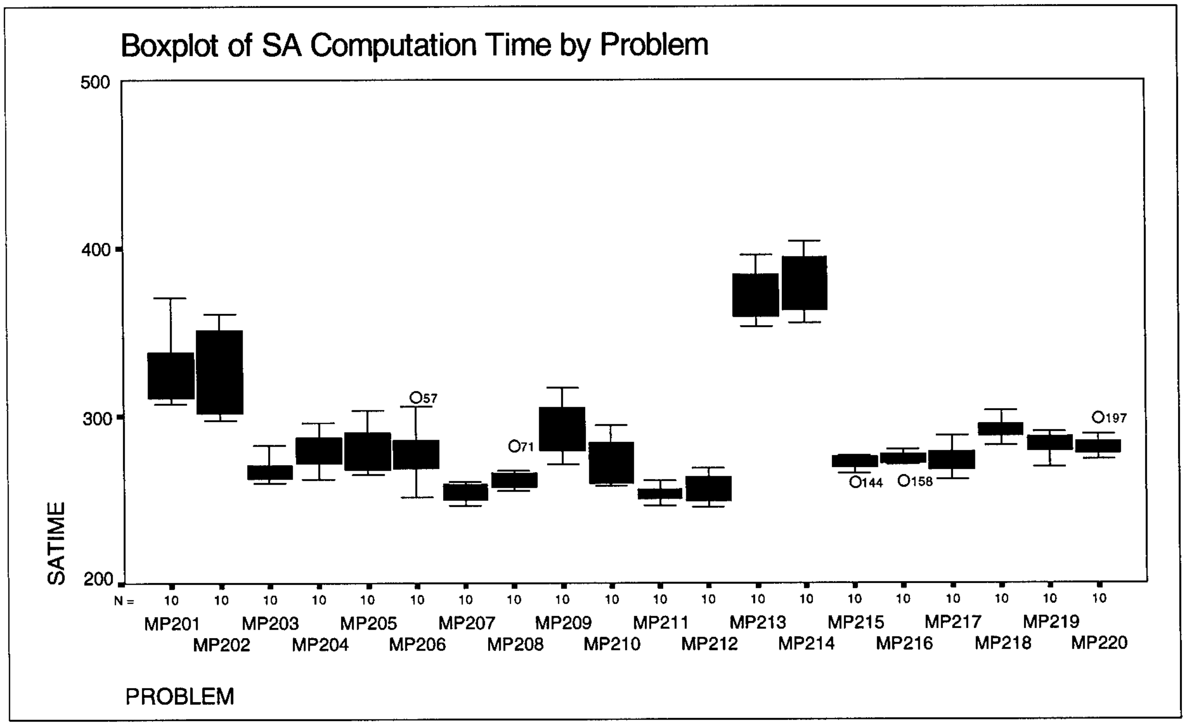

Figure 33. Boxplot of SA Computation Time by Problem (MP-FLP)

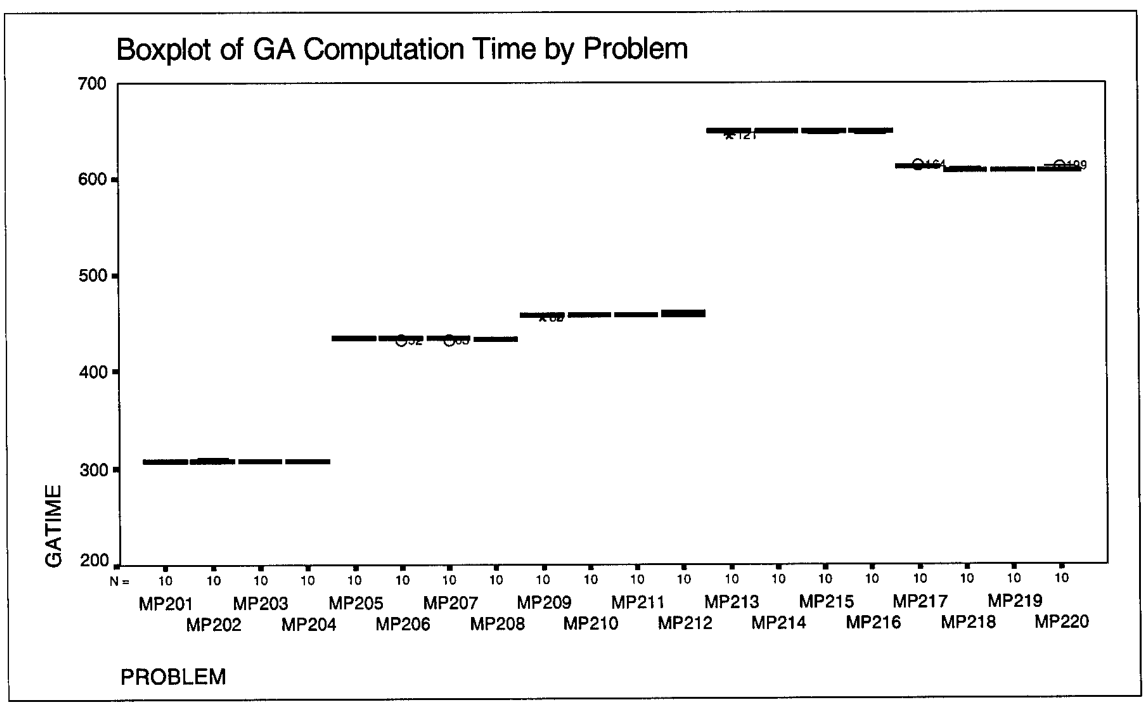

Figure 34. Boxplot of GA Computation Time by Problem (MP-FLP)

6.12 Chapter Summary

In this chapter we presented the implementation of three general heuristics for the multiple-period facilities location problem. We reported experimental results for selecting an appropriate problem representation. Our experiments indicated that a matrix representation gave the best results in terms of solution quality and time performance for tabu search and genetic algorithms, with mixed results for simulated annealing. Using the matrix representation, we reported results of a parameter search experiment for each heuristic. Results indicate that most parameters have a significant impact on solution quality and computation time requiring careful selection for best results. We then reported the results of a benchmark experiment with 20 mid-sized problems that were first solved to optimality. Results showed simulated annealing had the lowest average optimality gap at 3.36%, followed by tabu search at 6.85%, with the genetic algorithm trailing far behind with an average optimality gap of 15.23%. We conclude the chapter with some observations about the effect of computation time on solution quality and the effect of problem size on computation time. All results are as expected, with longer computation resulting in better results, and larger problems requiring more computation time. Results of statistical tests are reported.

In the following chapter, we will address the multiple-commodity facilities location problem. In Chapter 8 we report our principal results comparing the general performance of these heuristics based on larger test problems.

Chapter 7

HEURISTICS FOR MULTIPLE-COMMODITIES FACILITIES LOCATION PROBLEMS

In the previous two chapters, we presented general heuristics for the capacitated facilities location problem (CLFP) and the multiple-periods facilities location problem (MP-FLP). This chapter parallels the previous two chapters in presenting tabu search, simulated annealing, and genetic algorithm heuristics for the multiple-commodities facilities location problem (MC-FLP). In the first section we introduce the MC-FLP, present a formulation, and mention some of the relevant literature. In the following sections, we present the heuristic algorithms. For each one we specify our particular implementation and parameters. Since much of this work similar to the work done for CFLP, we make references to section in Chapter 5 whenever it is appropriate. After all three heuristics have been explained, we report the results from the first three stages of experimentation: selection of appropriate representation, parameter experimentation, and benchmarking.

7.1 The Multiple-Commodities Facilities Location Problem

For MC-FLP, M customers have demand for one or more W commodities. Our objective is to determine which of N capacity-constrained potential locations should supply which commodities, such that each location supplies only one kind

of commodity while minimizing the overall sum of fixed and variable costs. Warszawski (1973) proposed a branch and bound solution for this problem when locations have no capacity limitations. Neebe and Khumawala (1981) present an improved branch and bound algorithm with stronger bounds, but also for the uncapacitated version. Geoffrion and Graves (1974) solved a capacitated version using Bender's decomposition. Their formulation, however, allows each location to provide multiple commodities, while limiting each customer to a single source of supply. Our formulation is a capacitated version of the Warszawski problem:

$$\min Z = \sum_{r=1}^{W}\sum_{i=1}^{N}\sum_{j=1}^{M} C_{rij}x_{rij} + \sum_{r=1}^{W}\sum_{i=1}^{N} F_{ri}y_{ri} \tag{18}$$

subject to

$$\sum_{i=1}^{N} x_{rij} = D_{rj} \qquad j = 1,2,\ldots,M; r = 1,2,\ldots,W \tag{19}$$

$$\sum_{r=1}^{W} y_{ri} = 1 \qquad i = 1,2,\ldots,N \tag{20}$$

$$\sum_{j=1}^{M} x_{rij} \le S_i y_{ri} \qquad i = 1,2,\ldots,N; r = 1,2,\ldots,W \tag{21}$$

$$y_{ri} \in \{0,1\} \qquad i = 1,2,\ldots,N; \quad r = 1,2,\ldots,W \tag{22}$$

$$x_{rij} \ge 0 \qquad i = 1,2,\ldots N; \quad j = 1,2,\ldots,M; \quad r = 1,2,\ldots,W \tag{23}$$

where

D_{rj}	customer *j*'s demand for commodity *r*
S_i	location *i*'s capacity
C_{rij}	unit cost to satisfy customer *j*'s demand for commodity *r* from location *i*
F_{ri}	fixed cost of supplying commodity *r* from location *i*
x_{rij}	quantity of commodity *r* for customer *j* shipped from location *i*
y_{ri}	{0,1} indicator where $y_{ri}=1$ implies commodity *r* is supplied from location *i*

Specifically, fixed costs (F_i), variable costs (C_{rij}), location capacities (S_i), and customer demands (D_{rj}) are the given input data for the problem and the demand allocations (x_{rij}) and facility locations (y_{rij}) are the decision variables.

7.2 Solution Representations

As previously discussed, tabu search, simulated annealing, and genetic algorithms require an abstract representation of the problem's solution in order to define "moves" and "operators." Unlike CFLP and MP-FLP where we were able to propose alternate representations, MC-FLP can best be represented by a vector representation. Attempts to define an alternate matrix oriented representation failed due to the special constraint that each location may only supply a single commodity. This particular constraint makes defining neighborhood moves impractical for a matrix representation. Accordingly, we now describe our implementation of the vector representation.

The vector representation is indirectly linked to the y_{ri}'s in the problem formulation. A particular solution is represented by a vector $\mathbf{s} = \{y_1, y_2,\ldots,y_N\}$, where $y_i \in \{0, 1,\ldots,W\}$ indicating what commodity, if any, is supplied by location i (a zero value indicates the location is not in operation). Using this vector solution, the problem is partitioned into W single-commodity location problems. The x_{rij}'s are determined by solving the corresponding transportation problems for each single-commodity problem. The total cost is then easily computed using (18) above.

7.3 A Tabu Search Heuristic for MC-FLP

Previously, we defined a tabu search heuristic as the following set of elements:

$$\text{TS} = \{ \mathscr{R}, \mathscr{N}(), \mathscr{C}(), \mathscr{A}(), \mathscr{T}(), \mathscr{P} \}$$

Where

$\mathscr{R}$	is the solution representation structure
$\mathscr{N}()$	is the neighborhood structure;
$\mathscr{C}()$	is the cost function
$\mathscr{A}()$	is the aspiration criterion function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

In the previous section we discussed the solution representation and associated cost function. The aspiration criteria, terminating condition function, and control parameters are the same as for CFLP, which are described in §5.3.2, §5.3.3, and §5.3.4. It only remains to describe the neighborhood structure.

Given a vector representation, a neighbor solution is defined as a solution where the commodity assignment of one of the vector elements is changed to some other commodity or to a closed status (no commodity assigned). The complete neighborhood for an incumbent solution then is the set of all solutions resulting from changing the commodity assignment of each potential location from its current assignment to one of the other commodities in turn or to no commodity assignment. For a problem with w commodities and n potential locations, the neighborhood size is wn. Although this is not a very large neighborhood, having to solve transportation problems for each neighbor is a slow process, especially as the problems grow in size. Therefore, we only evaluate a portion of this neighborhood. At each iteration, we only evaluate the n neighbor solutions resulting from changing the current assignment of each

potential location in turn to some other commodity (or no commodity). The new commodity assignment is selected at random. In some cases, this operation will result in neighbor solutions that are not feasible due to insufficient capacity assigned to a particular commodity to meet all demand. In such cases, the resulting neighbor solution is ignored (i.e., no attempt is made to evaluate n feasible solutions, just n neighbor solutions, feasible or not).

7.4 A Simulated Annealing Heuristic for MC-FLP

In Chapter 2, we defined a simulated annealing algorithm as the following set of elements:

SA = { $\mathcal{R}, \mathcal{N}(), \mathcal{C}(), \mathcal{S}(), \mathcal{T}(), \mathcal{P}$ }

where

$\mathcal{R}$	is the solution representation structure
$\mathcal{N}()$	is the neighborhood structure;
$\mathcal{C}()$	is the cost function
$\mathcal{S}()$	is the cooling schedule
$\mathcal{T}()$	is the terminating condition function
$\mathcal{P}$	is the set of control parameters

Solution representation and the cost function were described in §7.2. The cooling schedule, terminating condition function, and control parameters are the same as for CFLP and are described in §5.4.2, §5.4.3, and §5.4.4. We now describe the neighborhood structure.

As for CFLP and MP-FLP, the neighborhood structure for simulated annealing is the same as for tabu search described in the previous section. In simulated annealing, given an incumbent solution, we randomly select one of the wn neighboring solutions to evaluate next. The neighbor solution is selected by randomly choosing one of the potential locations and changing its current commodity assignment to some other, randomly selected commodity (or no

commodity). If the resulting neighbor solution is not feasible, we simply try another one. Although neighbor solutions that are not feasible are ignored, they do count toward the total number of solutions considered with respect to the FACTOR parameter. This is a necessary condition to avoid lengthy searches for feasible solutions in highly constrained problems, especially toward the end of the simulated annealing algorithm, when the probability of accepting solutions is very small and the FACTOR parameter starts to take effect.

7.5 A Genetic Algorithm Heuristic for MC-FLP

In Chapter 2, we defined a genetic algorithm as the following set of elements:

$$GA = \{ \mathscr{R}, \mathscr{F}(), \mathscr{X}(), \mathscr{M}(), \mathscr{T}(), \mathscr{P} \}$$

Where

$\mathscr{R}$	is a solution representation
$\mathscr{F}()$	is the fitness function
$\mathscr{X}()$	is the crossover operation function
$\mathscr{M}()$	is the mutation operation function
$\mathscr{T}()$	is the terminating condition function
$\mathscr{P}$	is the set of control parameters

The solution representation is discussed in §7.2. The fitness function, terminating condition, parameters are the same as for CFLP and are discussed in §5.5.1, §5.5.4, and §5.5.5. We now describe the crossover and mutation operators.

7.5.1 Crossover Operator

We implement a single-point crossover operator on the solution vector representation. Recall that for the vector representation, each element represents a potential location, and the element's value represents the location's current commodity assignment. Given two parent chromosomes and a single crossover point (i.e., a position in the vector), an offspring solution receives the

commodity assignments from the first parent for locations below the crossover point and the commodity assignments from the second parent for locations at the crossover point and above. Since this operation may result in solutions that are not feasible, a repair operation is implemented.

Following the application of crossover and mutation operations, each new offspring is put through the repair operation. The repair operation iterates through each commodity, and checks the solution for sufficient capacity to meet demand. When this procedure finds a commodity with insufficient capacity assigned in the current solution, it goes into repair mode. At first, the repair operation consists of randomly selecting elements in the solution vector, looking for one with no commodity assignment. If one is found, then it is assigned the commodity that requires more capacity. This procedure continues until enough capacity has been added to the current commodity or a fixed number of attempts have elapsed without success (the number of attempts is equal to the number of potential locations). If the above procedure failed to assign enough capacity to the commodity with a capacity deficiency, the repair operation continues by randomly selecting elements of the solution vector, and determining if changing its current commodity assignment would reduce the total capacity assigned to this commodity below its required minimum. If not, then the commodity assignment for this element is changed to the one we are trying to repair. This process continues until enough capacity has been reassigned to the commodity we are trying to repair and until all commodities have enough capacity assigned to them. Although this repair operation can drastically change the structure of an

offspring solution, it is needed to ensure all parent chromosomes in a new generation are feasible. However, in early tests we discovered that this operation is not often performed.

7.5.2 Mutation Operator

The mutation operator is applied probabilistically to an entire chromosome. The mutation consists of randomly selecting an element in the solution vector, and changing its current commodity assignment to some other, randomly selected commodity (or no commodity). This operation may result in solutions that are not feasible. As a result, the repair operation described above is applied after both the crossover and mutation operators are applied.

7.6 Benchmark Test Problems

A set of 20 medium-sized test problems was created for the purposes of parameter effects experimentation and benchmarking. The data for the test problems were obtained from a database of 250 U.S. cities with their inter-city distances and 1990 Census population (see Appendix C). The problem sizes selected were based on the problem sizes used by Neebe and Khumawala (1981). The set of 20 benchmark problems is summarized in Table 34. We now describe each of the test problem attributes:

Size: Neebe and Khumawala used some problems from Warszaski with 2 commodities x 6 locations x 10 customers, 3x6x12, 3x7x12, and 3x9x38. They also used larger problems based on the data of Khuen and Hamburger: 3x20x50, 3x21x50, 3x25x50, 3x26x50, 4x10x50, 4x11x15, 4x15x50, and 4x16x50. Of

Table 34. Benchmark Test Problems (MC-FLP)

Problem	Size	Capacity	Fixed Costs
MC201	3x15x25	20000	7500
MC202	3x15x25	20000	15000
MC203	3x15x25	40000	7500
MC204	3x15x25	40000	15000
MC205	3x20x50	35000	7500
MC206	3x20x50	35000	15000
MC207	3x20x50	70000	7500
MC208	3x20x50	70000	15000
MC209	3x25x50	30000	7500
MC210	3x25x50	30000	15000
MC211	3x25x50	60000	7500
MC212	3x25x50	60000	15000
MC213	4x10x50	100000	7500
MC214	4x10x50	100000	15000
MC215	4x10x50	200000	7500
MC216	4x10x50	200000	15000
MC217	4x15x50	65000	7500
MC218	4x15x50	65000	15000
MC219	4x15x50	130000	7500
MC220	4x15x50	130000	15000

these, we selected the following five sizes: 3x20x50, 3x25x50, 4x10x50, 4x15x50, and we also added a smaller 3x15x25.

Capacity: as for MP-FLP, we set capacity at two levels. A low level assigns enough capacity to each potential location such that the total capacity of all locations is twice the total demand for all customers for all commodities. A high level doubles this capacity assignment. Note that for a given problem, all locations receive the same capacity assignment.

Demand: the demand for the first commodity is a fraction of the customer city's 1990 Census population. The demands for the remaining commodities are a random factor of the first commodity's demand. However, instead of selecting a single factor for each commodity which would result in each commodity having the same distribution of demand over all customers, each commodity-customer combination gets a separate random factor, thereby changing the distribution of

demand for each commodity across customers. The random factor is a number ranging from 0.5 to 1.5.

Fixed Costs: Two levels of fixed costs were assigned, 7500 and 15000. For a given problem, all locations receive the same fixed cost. The fixed cost does not change based on the commodity assigned to a location.

Variable Costs: a base variable cost is assigned to each source-destination combination based on the distance between the two cities and charged at 2.5 cents per mile. The actual variable cost used in computing solution costs is based on a factor randomly assigned to each commodity. The factor is a number between 0.5 and 1.5. The effective variable cost for a particular commodity's source-destination cell is then the product of the base variable cost and the appropriate factor.

7.7 Parameter Effects

As we did for CFLP and MP-FLP, we conducted an experiment to get a clear picture of how various parameter settings affect solution quality and computation time. We also use the results from these experiments to select the parameter settings that we use for benchmarking and comparative runs.

For tabu search, we experimented with three levels of the LTM parameter, four levels of the STM parameter, and three levels for the TLSIZE parameter. Compared with the levels used for CFLP and MP-FLP, the experimental levels for MC-FLP are considerably more conservative due first to the fact that the vector representation requires solving transportation problems and is thus much

slower per iteration, but also because the solution space is much smaller than for a matrix representation. The experimental levels are summarized in Table 35.

Table 35. TS Parameters Experimental Levels (MC-FLP)

Parameter	Experimental Levels			
LTM	1	3	5	
STM	25	50	75	100
TLSIZE	1	2	5	

For simulated annealing, we experimented with five levels for the RATE parameter, three levels for the ACCEPT parameter, and three levels for the FACTOR parameter. The experimental levels are summarized in Table 36.

Table 36. SA Parameters Experimental Levels (MC-FLP)

Parameter	Experimental Levels				
RATE	0.75	0.80	0.85	0.90	0.95
ACCEPT	25	50	75		
FACTOR	1	2	3		

For genetic algorithm we experimented with three levels for the POP parameter, three levels for the GENS parameter, three levels for the XOVER parameter, and two levels for the MUTATE parameter. The experimental levels are summarized in Table 37.

Table 37. GA Parameters Experimental Levels (MC-FLP)

Parameter	Experimental Levels		
POP	20	30	40
GENS	50	100	150
XOVER	0.80	0.85	0.95
MUTATE	0.05	0.10	

To conduct these experiments, we selected one of the benchmark test problems as a test bed. Since the results from the parameter experiments lead

to a selection of parameters for solving the set of larger problems, we selected one of the largest benchmark problems, MC209. Although this problem only had three commodities, early experiments indicated that the speed at which the heuristics perform is more closely linked to the number of locations and customers than to the number of commodities. This is a direct result of having to solve transportation problems. With 25 locations and 50 customers, MC209 is one of the largest problems, which also has a tight capacity constraint, a factor we have observed makes the problems more difficult to solve.

The tabu search experiment has 36 cells, the simulated annealing experiment has 45 cells, and the genetic algorithm experiment has 54 cells. As we have done for CFLP and MP-FLP, we limited each experimental cell to 10 replications.

To determine if there is a statistically significant difference among the experimental levels for each parameter with respect to solution quality and computation time, we computed the Kruskal-Wallis (K-W) statistic. To determine the general direction of the effect and its statistical significance, we also computed Spearman's rank correlation coefficient for each parameter effect. The results of these statistical tests are summarized in Tables 38, 39, 40, and 41 and are discussed in the following sections. The raw result tables can be found in Appendix B. Boxplots showing the effects can be found in Appendix D.

7.7.1 Tabu Search LTM Parameter

The effects of the LTM parameter on solution quality are evident from Figure 35. The effect is the same as for CFLP and MP-FLP as expected. There is a

Table 38. Kruskal-Wallis H Test of Heuristic Parameters Effect on Solution Quality (MC-FLP)

Parameter	Chi-Square	D.F.	Significance
LTM	124.5743	2	.0000
STM	0.8703	3	.8326
TLSIZE	0.4673	2	.7916
RATE	0.4767	4	.9757
ACCEPT	0.5258	2	.7688
FACTOR	0.4509	2	.7982
POP	13.2454	2	.0013
GENS	269.2745	2	.0000
XOVER	1.9637	2	.3746
MUTATE	40.5796	1	.0000

Table 39. Kruskal-Wallis H Test of Heuristic Parameters Effect on Computation Time (MC-FLP)

Parameters	Chi-Square	D.F.	Significance
LTM	214.9260	2	.0000
STM	132.6687	3	.0000
TLSIZE	0.0179	2	.9911
RATE	205.3205	4	.0000
ACCEPT	149.5801	2	.0000
FACTOR	75.8593	2	.0000
POP	99.7461	2	.0000
GENS	423.5083	2	.0000
XOVER	0.0959	2	.9532
MUTATE	2.0482	1	.1524

significant improvement on solution quality as the number of long-term memory passes is increased, particularly going from a single pass, to multiple passes. A K-W statistic of 124.5743 ($p = .0000$) confirms the significant improvement, and a correlation coefficient of -.5790 ($p = .000$) confirms the direction of this effect.

With respect to computation time, the effects of the LTM parameter are also clear and as expected (Figure 36). More passes of the long-term memory require longer computation time. The K-W statistic of 214.9260 ($p = .0000$) and correlation coefficient of .7657 ($p = .000$) confirm these observations.

Table 40. Spearman's Correlation Coefficients of Heuristic Parameters vs Solution Quality

Parameter	Coefficient	Significance
LTM	-.5790	.000
STM	-.0385	.467
TLSIZE	-.0138	.795
RATE	-.0324	.493
ACCEPT	-.0328	.488
FACTOR	-.0315	.504
POP	-.1509	.000
GENS	-.6931	.000
XOVER	-.0342	.0024
MUTATE	-.2744	.000

Table 41. Spearman's Correlation Coefficient of Heuristic Parameters vs Computation Time

Parameter	Coefficient	Significance
LTM	.7657	.000
STM	.6009	.000
TLSIZE	-.0029	.956
RATE	.6644	.000
ACCEPT	.5705	.000
FACTOR	.3999	.000
POP	.4300	.000
GENS	.8805	.000
XOVER	.0024	.955
MUTATE	.0616	.153

7.7.2 Tabu Search STM Parameter

As the STM parameter is increased, one would expect a downward slope on each line in Figure 35. Surprisingly, such a slope is not present. More importantly, the K-W test statistic of .8703 ($p = .8326$) indicates there is no significant difference in solution quality as a result of increasing the STM parameter and the correlation coefficient of -.0385 ($p = .467$) indicates no correlation between STM and solution cost. These results are a departure from what we have seen for CFLP and MP-FLP where the STM parameter had a significant effect.

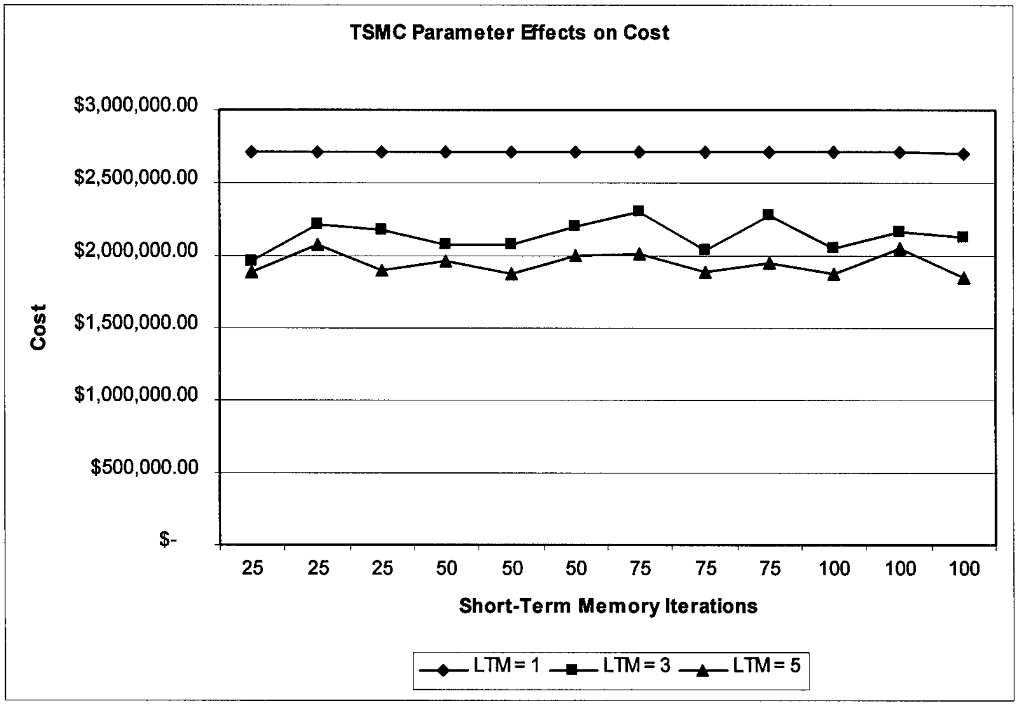

Figure 35. TS Parameters Effect on Cost (MC-FLP)

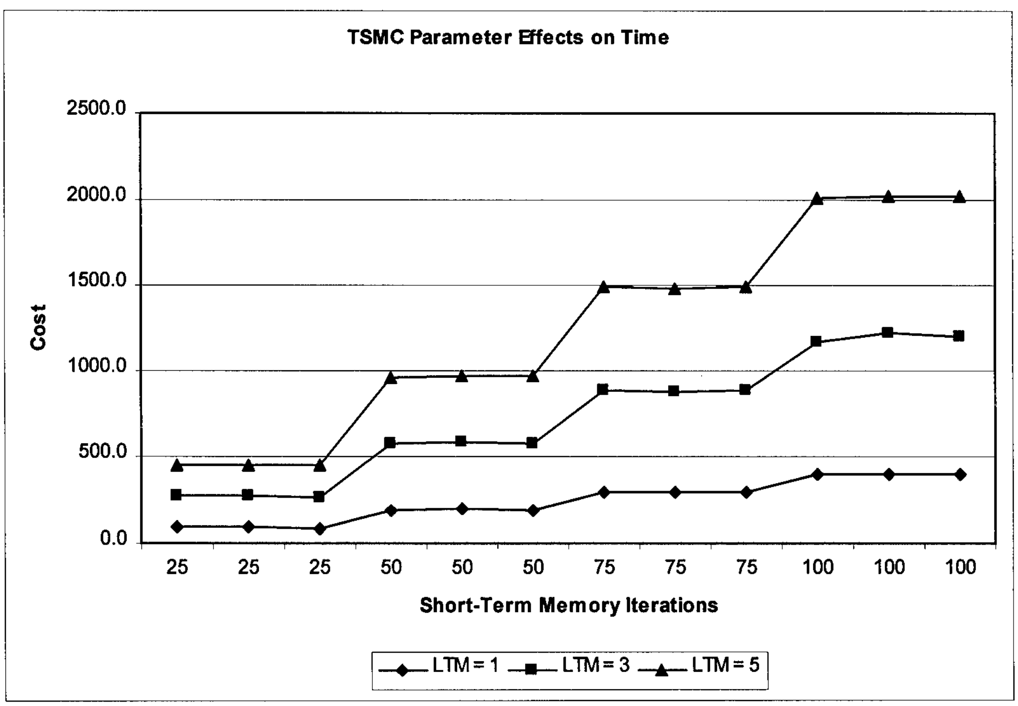

Figure 36. TS Parameters Effect on Time (MC-FLP)

The slope of each line in Figure 36 represents the effects of the STM parameter on computation time. The effects in this case are conventional and as expected. The K-W statistic of 132.6687 ($p = .0000$) and correlation coefficient of .6009 ($p = .000$) indicate a significant effect: more STM iterations require more computation time.

7.7.3 Tabu Search TLSIZE Parameter

The effects of TLSIZE can be seen in Figure 35 by looking at each of the lines, three points at a time, corresponding to the three levels of TLSIZE. For the line with a single long-term memory pass, TLSIZE has no effect on solution quality. For the other two lines the effect is mixed. A K-W statistic of .4673 ($p = .7619$) and a correlation coefficient of -.0138 ($p = .795$) indicate there is no significant impact on solution quality due to the TLSIZE parameter.

Looking at each line in Figure 36, it is obvious that each of the three point groupings on each line representing the three levels of the TLSIZE parameter do not have a significant impact on computation time. This is confirmed by a correlation coefficient of -.0029 ($p = .956$) and a K-W statistic of .0179 ($p = .9911$).

7.7.4 Tabu Search Parameters Selected

Based on the preceding results, we selected the following parameter settings for future experiments: LTM = 5, STM = 50, and TLSIZE = 2. A setting of STM = 100 improves the solution quality by 1.7%, but requires a 108% increase in computation time which cannot be justified.

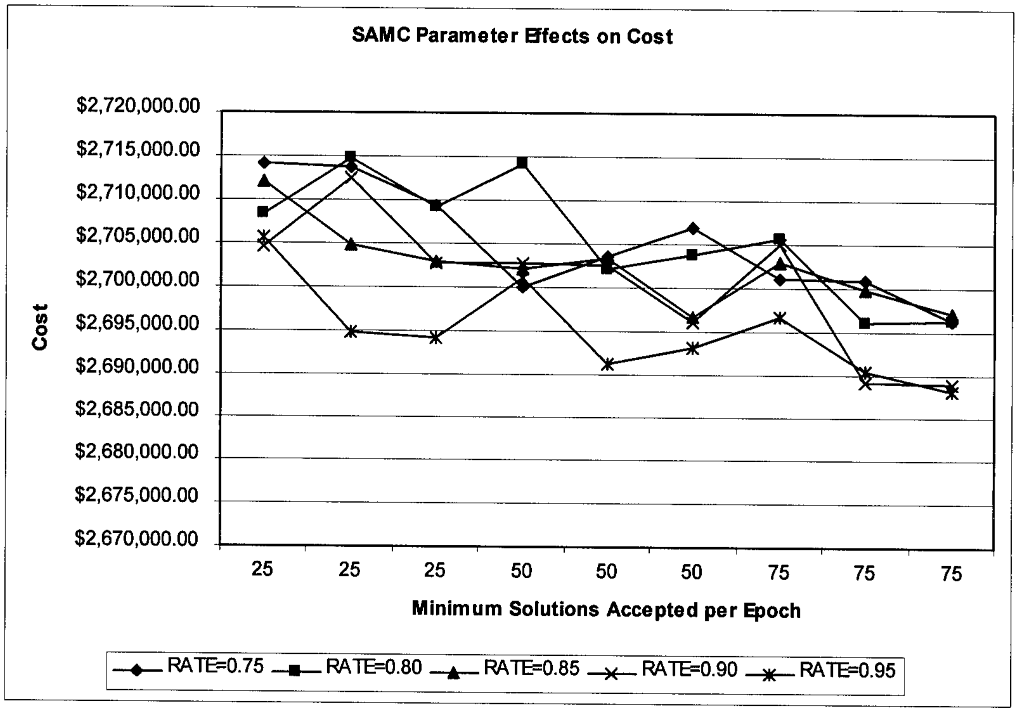

Figure 37. SA Parameters Effect on Cost (MC-FLP)

7.7.5 Simulated Annealing RATE Parameter

Looking at each line in Figure 37, the effects of the RATE parameter are not visibly clear. There is some degree of overlap among various levels of the RATE parameter. The statistical tests suggest that this parameter does not have a significant impact on solution quality (correlation coefficient of -.0324, p = .493 and K-W statistic of .4767, p = 9757) although it would seem that the 0.95 level gives the best results except for the first and last two points where there is some overlap with the 0.90 level.

If the RATE parameter does not have a significant effect on solution quality, its effect on computation time is quite clear from Figure 38. The higher RATE levels require more time due to a greater number of solutions evaluated. The

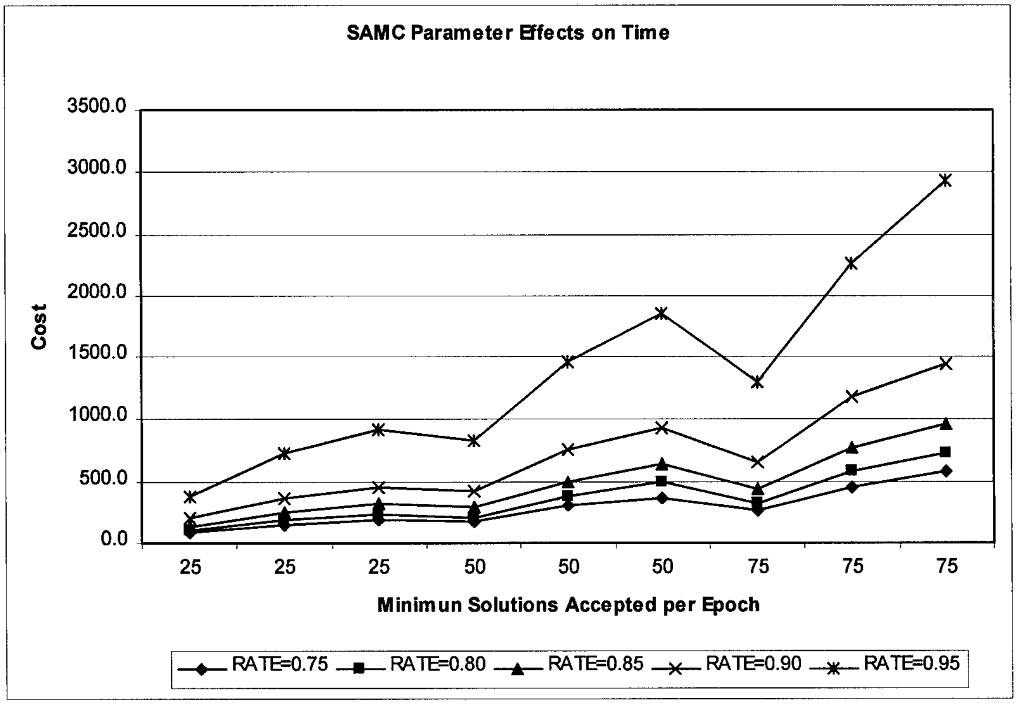

Figure 38. SA Parameters Effect on Time (MC-FLP)

statistical tests confirm this observation with a K-W statistic of 205.3205 (p = .0000) and a correlation coefficient of .6644 (p = .000).

7.7.6 Simulated Annealing ACCEPT Parameter

The expected effect of the ACCEPT parameter would be represented in Figure 37 as a downward slope for each line in the graph. As a whole, the lines do seem to have an overall downward slope, but individually, the effect is not consistent. The statistical tests indicate no significant effect by the ACCEPT parameter on solution quality. The correlation coefficient is -.0328 (p = 488) and the K-W test statistic is .5258 (p = .7688).

With respect to computation time, however, the effect of the ACCEPT parameter are conventional. The consistent upward slope of each line in Figure 38 indicates as expected that a higher level of the ACCEPT parameter requires

more computation time. A correlation coefficient of .5705 (p = .000) and K-W statistic of 149.5801 (p - .0000) confirm these observations.

7.7.7 Simulated Annealing FACTOR parameter

Trying to visualize the effects of the FACTOR parameter on solution quality by looking at Figure 37 is rather difficult. However, if we try to isolate the three point groupings for each line, we can see there is no consistent pattern across lines, and in some cases even within each line. This would suggest a mixed effect of the FACTOR parameter on solution quality. The statistical tests indicate the FACTOR parameter does not have a significant effect. The correlation coefficient is -.0315 (p = .504) and the K-W statistic is .4509 (p = 7982).

In the case of computation time, the effect of the FACTOR parameter is again quite clear from Figure 38. For each line, the three point groupings clearly show a consistent upward pattern suggesting the expected result that higher levels of the FACTOR parameter require more computation time. A correlation coefficient of .3999 (p = .000) and K-W statistic of 75.8593 (p = .0000) confirm this effect.

7.7.8 Simulated Annealing Selected Parameters

Based on the preceding results, we selected the following parameter settings for the remaining experiments: RATE = 0.90, ACCEPT = 50, and FACTOR = 3. Other settings improve solution quality by no more than 0.3% with a minimum increase in computation time of 26% and as much as 213%.

7.7.9 Genetic Algorithm POP parameter

Unfortunately, Figure 39 is difficult to interpret when trying to determine the effects of the POP parameter on solution quality. For the effect to be obvious, each of the lines in Figure 39 would have to show a consistent separation. While there is some separation at most points, there is also some overlap at others. The statistical tests, however, indicate there is in fact a significant effect. The correlation coefficient -.1509 ($p = .000$) shows the expected negative correlation and the K-W statistic of 13.2454 (.0013) indicates there is some significant difference among the experimental levels.

Figure 40 shows the expected effects of the POP parameter on computation time. Each pair of lines on this graph represents a level of the POP parameter, with the higher levels requiring more computation time. A correlation coefficient of .4300 ($p = .000$) and K-W statistic of 99.7461 ($p = .0000$) indicate a significant effect.

7.7.10 Genetic Algorithm GENS Parameter

The effect of the GENS parameter is evident in Figure 39 by the slope of each line. As expected, more generations result in improved solution quality. A correlation coefficient of -.6931 ($p = .000$) indicates a strong negative correlation as expected. A K-W statistic of 269.2745 ($p = .0000$) indicates a significant difference among the experimental levels.

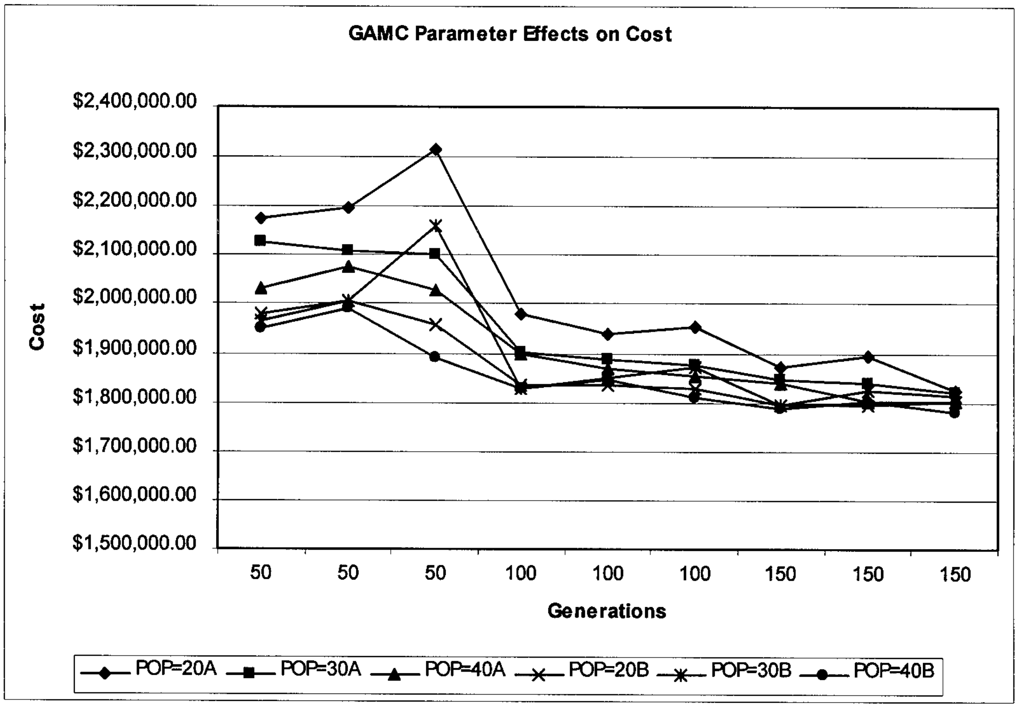

Figure 39. GA Parameters Effect on Cost (MC-FLP)

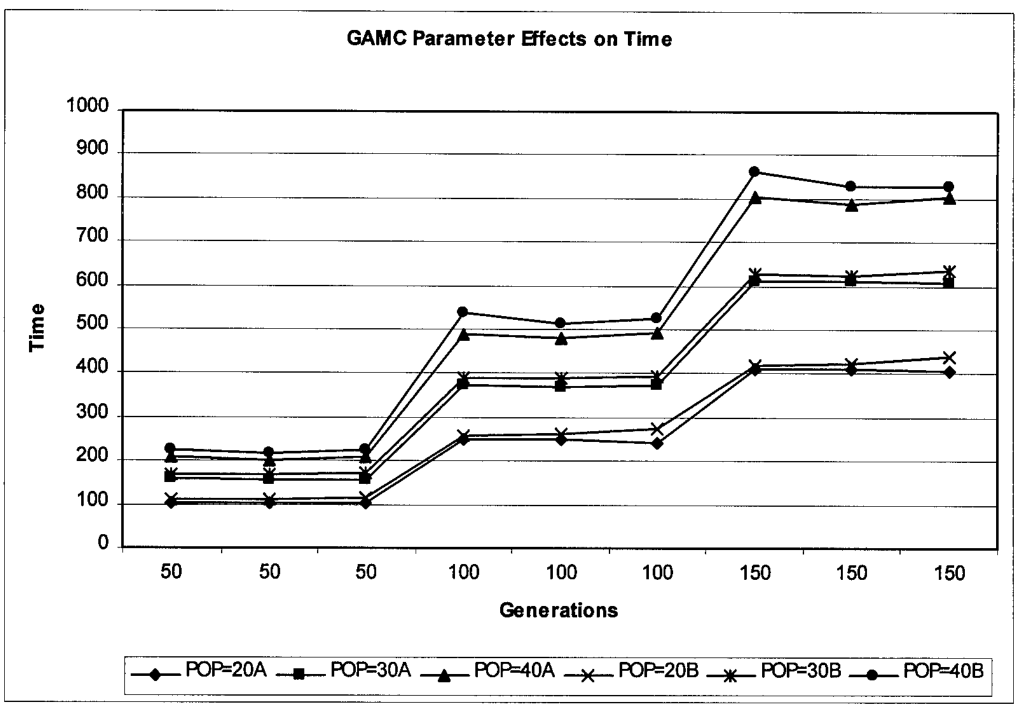

Figure 40. GA Parameters Effect on Time (MC-FLP)

With respect to computation time, the effect of the GENS parameter is as expected as seen in Figure 40. More generations require more computation time resulting from more solutions being evaluated. A correlation coefficient of .8805 ($p = .000$) and K-W statistic of 423.5083 ($p = .0000$) confirm this observation.

7.7.11 Genetic Algorithm XOVER Parameter

Again, the effects of the XOVER parameter are difficult to interpret from Figure 39. However, if we isolate the three point groupings for each line on this graph, we can see that there is no consistent pattern to the groupings suggesting a mixed effect tied to the settings of the other parameters. The statistical tests give the same results as for CFLP and MP-FLP: no significant effect on solution quality from the XOVER parameter. The correlation coefficient is -.0342 ($p = .427$) and the K-W statistic is 1.9637 ($p = .3746$).

Figure 40 shows the effects of the XOVER parameter on computation time. Looking at the three point groupings for each line in the graph, we see there is not much impact, if any, on time. The correlation coefficient of .0024 ($p = .955$) and K-W statistic of .0959 ($p = .9532$) indicate there is no significant effect.

7.7.12 Genetic Algorithm MUTATE Parameter

Whereas the effects of the MUTATE parameter were easily visible on a graph for CFLP and MP-FLP, it takes a little more effort to visualize this effect in Figure 39. If we isolate the two groups of lines corresponding to the two levels of the MUTATE parameter (marked “A” and “B” in the legend), we can see that in general, the lines marked with “A” in the legend are above the lines marked with “B” in the legend. This is a remarkable result since it is a complete reversal of

the results we have seen for CFLP and MP-FLP. The better performing lines marked "B" represent a 10% mutation rate. This is the level of MUTATE that did not perform well for CFLP and MP-FLP. Clearly, the change in representation from a matrix to a vector must be correlated to the effect of the MUTATE parameter. Furthermore, the effects of the MUTATE parameter are significant with a correlation coefficient of -.2744 (p = .000) and K-W statistic of 40.5796 (p = .0000).

The effect of the MUTATE parameter on computation time can be seen in Figure 40. Each pair of lines represents a level of the POP parameter, and each line in a pair represents one of the two levels of the MUTATE parameter. A visual inspection would seem to indicate there is a slight separation between each line in a pair. The statistical tests do not show a significant effect, however. The correlation coefficient is .0616 (p = .153) and the K-W statistic is 2.0482 (p = .1524).

7.7.13 Genetic Algorithm Selected Parameters

Based on the preceding results, we selected the following parameter settings for the remaining experiments: POP = 40, GENS = 150, XOVER = 0.95, and MUTATE = 0.10. If we look at Table 109 in Appendix B, we can see that these settings give the best results.

7.8 Benchmark Test Results

In this section we report the results from benchmarking experiments. We used the 20 benchmark test problems described earlier in Table 34. As before, we tried to use an optimal algorithm to solve each of these problems to

optimality. Unfortunately, the branch and bound algorithm we developed based on the work of Warszawski (1973) and Neebe and Khumawala (1981) failed to solve these problems to optimality. This is probably due to the fact the earlier algorithms were developed for uncapacitated problems, while ours is for capacitated problems and we were unable to use most of their "bound" arguments. Accordingly, we allowed the algorithm to run for over six hours, and used their best upper bound as an alternate benchmark. For two of the problems noted in Table 43, the algorithm failed to find an upper bound after six hours.

Table 42. Parameter Settings for Benchmark Experiments (MC-FLP)

TS Parameters		SA Parameters		GA Parameters	
LTM	5	RATE	0.90	POP	40
STM	50	ACCEPT	50	GENS	150
TLSIZE	2	FACTOR	3	XOVER	0.95
				MUTATE	0.10

Following our attempt to solve each problem to optimality, we solved each problem using each of the general heuristics described in this chapter. Parameter settings were selected based on the results from the previous parameter experiments and are summarized in Table 42. For each problem, we completed ten replications with each heuristic. The average results and resulting optimality gaps are reported in Table 43. Note that in some cases, our heuristic solutions are much better than the best upper bound found by branch and bound after six hours.

Table 43. MC-FLP Benchmark Results

	Tabu Search			Simulated Annealing			Genetic Algorithm		
Problem	Solution	Opt Gap	Time	Solution	Opt Gap	Time	Solution	Opt Gap	Time
MC201	$2,513,218	6.3%	60	$2,790,880	18.0%	138	$2,374,076	0.4%	121
MC202	$2,593,172	3.0%	52	$2,882,918	14.5%	112	$2,471,121	-1.9%	113
MC203	$2,707,595	15.3%	56	$3,405,018	44.9%	135	$2,487,104	5.9%	104
MC204	$2,705,663	8.3%	52	$3,489,782	39.7%	101	$2,597,730	4.0%	94
MC205	$3,282,077	-32.7%	481	$3,709,011	-23.9%	826	$3,075,611	-36.9%	676
MC206	$3,399,064	-31.5%	469	$3,852,876	-22.4%	804	$3,214,873	-35.2%	626
MC207	$3,356,728	-29.3%	377	$3,916,968	-17.5%	623	$3,114,623	-34.4%	469
MC208	$3,453,437	-28.6%	360	$4,065,350	-15.9%	669	$3,504,498	-27.5%	456
MC209	$1,877,646	*	713	$2,696,057	*	935	$1,779,199	*	838
MC210	$2,203,418	*	663	$2,861,598	*	902	$1,982,141	*	807
MC211	$2,002,265	-46.7%	578	$3,305,282	-12.0%	769	$1,873,846	-50.1%	597
MC212	$2,418,451	-37.3%	539	$3,460,361	-10.2%	741	$2,053,826	-46.7%	558
MC213	$7,528,714	2.7%	53	$7,825,085	6.7%	182	$7,411,905	1.1%	293
MC214	$7,592,804	2.5%	53	$7,904,456	6.7%	193	$7,461,155	0.7%	294
MC215	$7,848,437	7.8%	80	$9,902,772	36.1%	239	$7,803,713	7.2%	214
MC216	$7,883,850	7.2%	79	$9,978,080	35.7%	268	$7,871,203	7.0%	223
MC217	$5,561,852	-3.6%	227	$6,201,022	7.5%	495	$5,537,150	-4.0%	430
MC218	$5,718,910	-2.2%	226	$6,302,366	7.7%	500	$5,651,483	-3.4%	432
MC219	$5,702,113	-0.3%	205	$6,509,843	13.8%	430	$5,712,703	-0.1%	331
MC220	$5,806,138	0.6%	204	$6,631,819	14.9%	452	$5,792,986	0.4%	337

* Branch and bound algorithm failed to find an upper bound for these problems

For the benchmark experiments in Table 43, we did not compute the average optimality gap as we have done for CFLP and MP-FLP as we have some missing values and some cases where the heuristics performed better than the branch and bound upper bound. However, unlike CFLP and MP-FLP, the genetic algorithm is clearly the best performer, while the simulated annealing heuristic is a distant third for MC-FLP. This preliminary observation may lead to some interesting findings as we experiment with larger test problems and measure the heuristics' performances.

7.9 Computation Time vs. Solution Quality

To determine what kind of relationship exists between computation time and solution quality we again use the data collected during the parameter selection

experiments. Tables 39 and 41 indicate that most parameters have a significant impact on computation time. The immediate by-product of these experiments is a set of data points with a range of computation times for the same problem with each heuristic. Thus, we can use Spearman's rank correlation coefficient to determine if there is a relationship between solution time and solution quality.

The correlation coefficients summarized in Table 44 below indicate that for tabu search and genetic algorithm there is a significant negative correlation. In the case of simulated annealing, no significant correlation is detected. This result for simulated annealing parallels the results in Tables 38 and 40 indicating no significant parameter effects on solution quality.

Table 44. Spearman's Rank Correlation Coefficients for Computation Time vs. Solution Quality (MC-FLP)

Heuristic	Spearman's	Significance
Tabu Search	-.4996	.000
Simulated Annealing	-.0255	.589
Genetic Algorithm	-.7329	.000

In order to appreciate how these heuristics perform over time, we again show a time performance graph in Figure 41. The time performance of the heuristics is for problem MC209, the problem used for parameter experimentation, with parameters set as for the benchmark experiments shown in Table 42.

Figure 41 shows that with a vector representation, the time performance of these general heuristics changes drastically from what we have seen in Figures 21 and 31 for CFLP and MP-FLP. First, since fewer solutions are evaluated by each heuristic, there are fewer opportunities for improvements. This is evidenced in Figure 41 by the fewer number of points plotted in comparison to

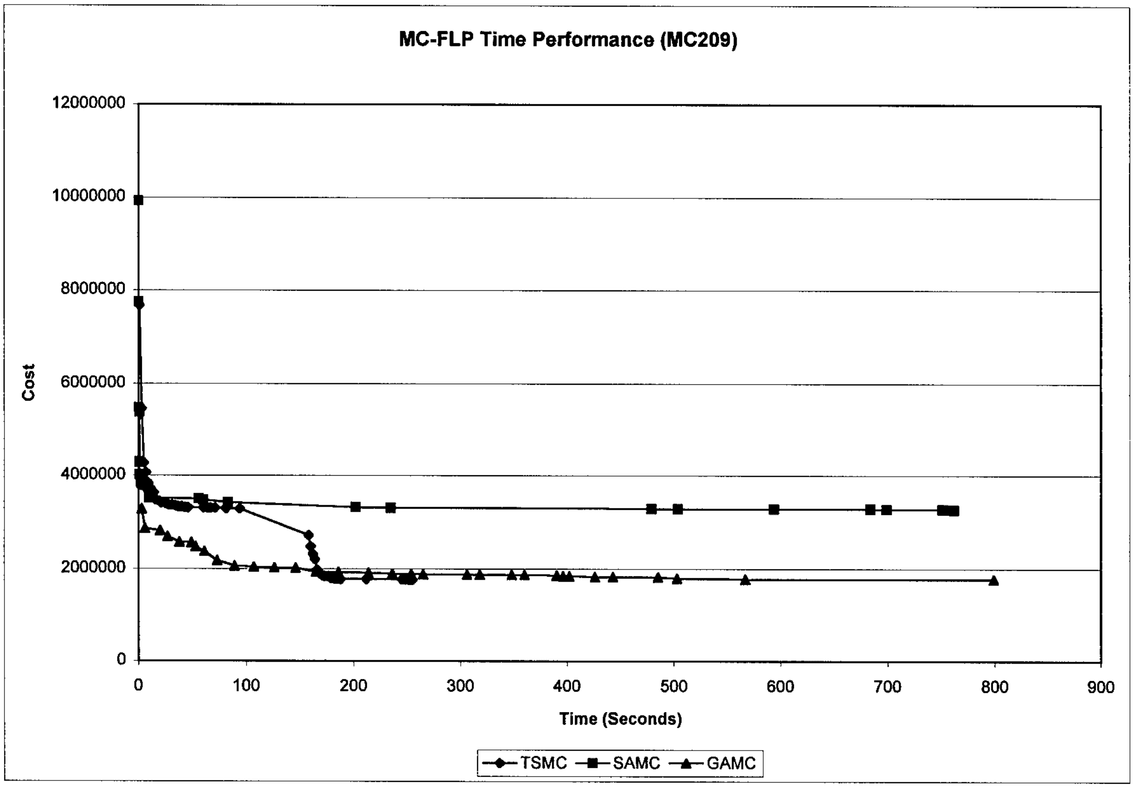

Figure 41. Heuristics Time Performance (MC-FLP)

the other two earlier figures. Second, we see that the genetic algorithm is now the one that finds better solutions faster. As we experiment with larger problems in the next chapter, we will address this finding further.

7.10 Problem Size vs. Computation Time

To examine the relationship between problem size and computation time we use the data collected during the benchmarking experiments. These data were collected from four problems for each of five sizes as described in Table 34. We coded each problem size with a number from one through five according to the number of locations and customers (because of the need to solve transportation problems, the number of locations and customers are the principal factors in computation time). Problems MC201-MC204 were coded as 1; problems

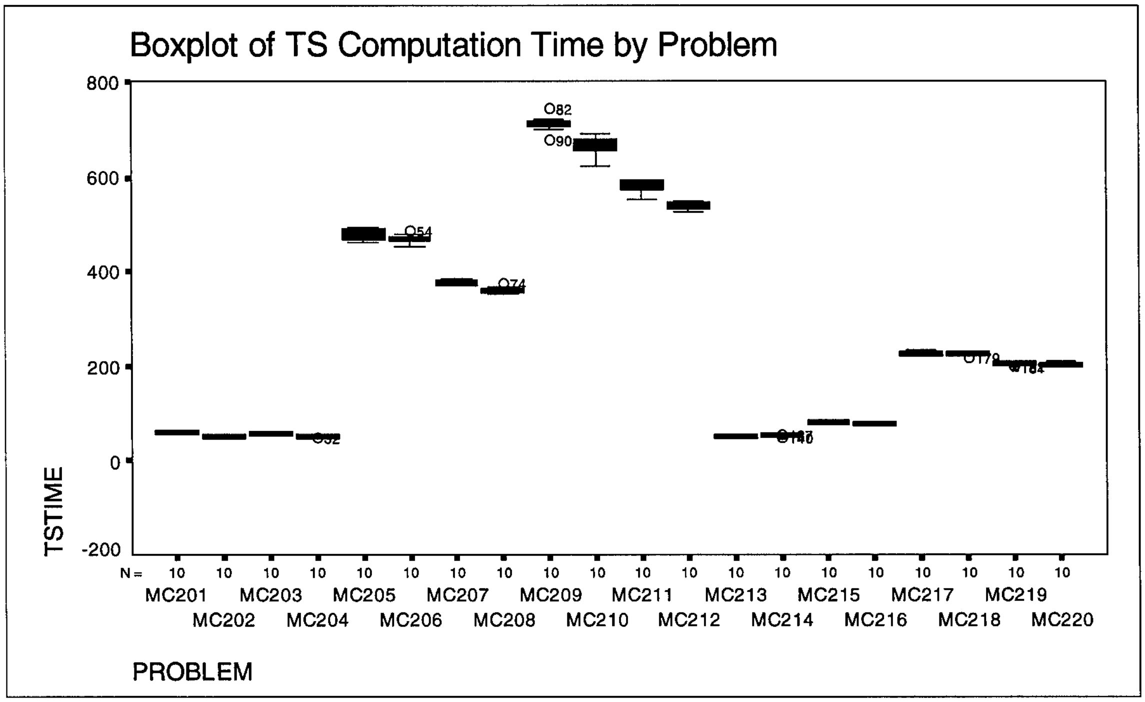

Figure 42. Boxplot of TS Computation Time by Problem (MC-FLP)

MC205-MC208 were coded as 4; problems MC209-MC212 were coded as 5; problems MC213-MC216 were coded as 2; and, problems MC217-MC220 were coded as 3. The rationale for this coding is obvious from the boxplot charts in Figures 42, 43, and 44.

As expected, the results indicate that problems with more locations and customers require more computation time. Tabu search had a correlation coefficient of .9476, simulated annealing had a coefficient of .9505, and the genetic algorithm had a coefficient of .9557. All had p = .000.

Looking at Figure 42, we notice that tabu search has little computation time variability due to different random number streams. Further, we notice that the problems that took the longest to compute are in fact the ones with more

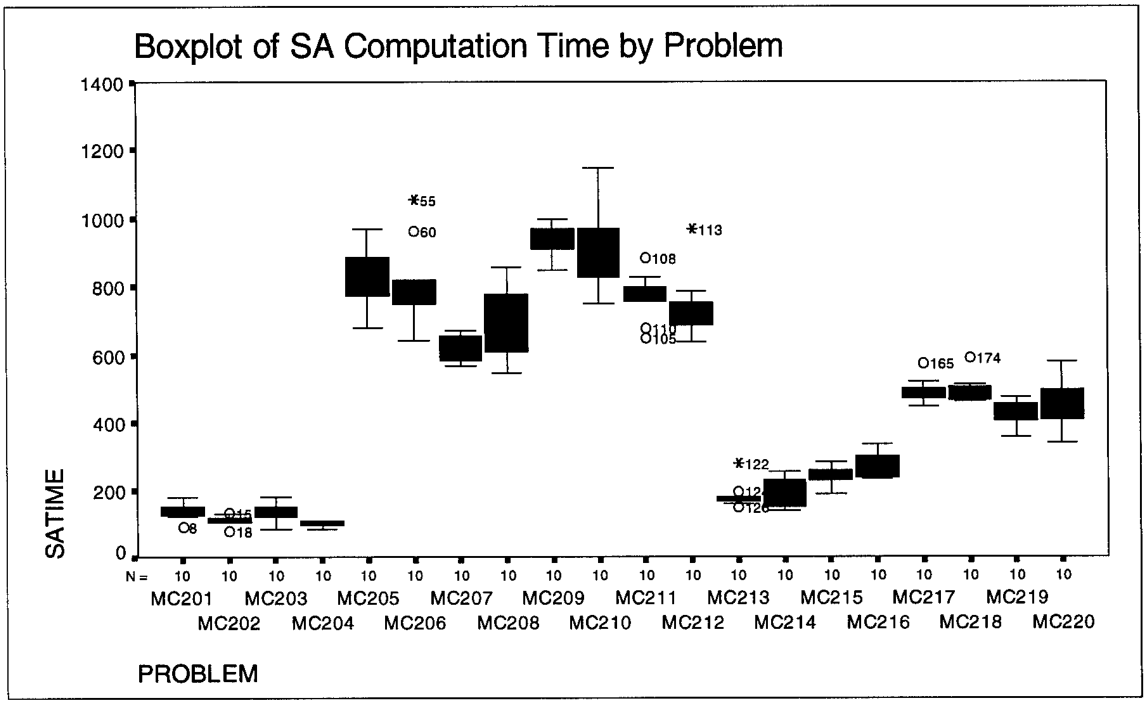

Figure 43. Boxplot of SA Computation Time by Problem (MC-FLP)

locations and customers, the second and third sets with 3x20x25 and 3x25x50 dimensions.

Figure 43 shows that simulated annealing exhibits more computation time variability, but also has the same type of distribution as tabu search. Again, the second and third sets of problems are the ones that take the longest.

Finally, Figure 44 shows that the computation time requirements have the same distribution along problem sizes as for the other two heuristics. It also shows for the second and third set the effect of the capacity constraint on computation time. Problems MC209, MC210, MC213, and MC214 have a tight capacity constraint and require more time than the other four problems in these two sets, which have double the capacity. This effect is also evident in the other two figures, though not as clearly.

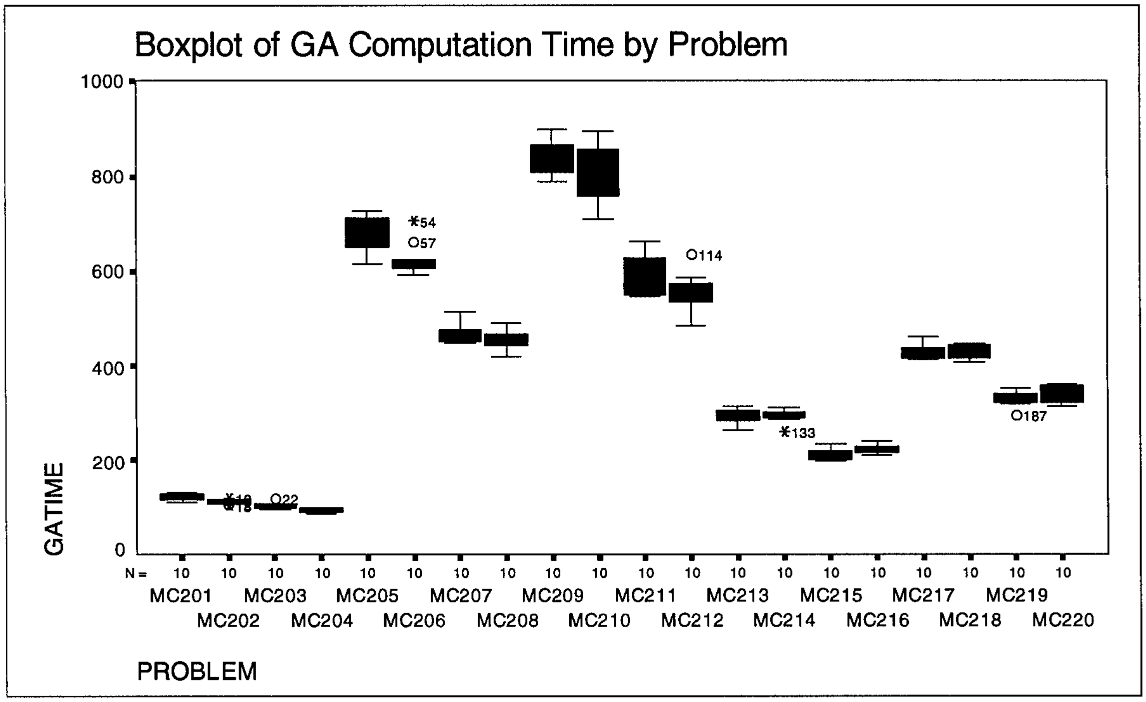

Figure 44. Boxplot of GA Computation Time by Problem (MC-FLP)

7.11 Chapter Summary

In this chapter we presented the development of three general heuristics for the multiple-commodity facilities location problem. We reported results of a parameter search experiment using a vector problem solution representation. Results indicate that for tabu search and genetic algorithm, most parameters have a significant effect on solution quality. This was not the case for simulated annealing. With respect to computation time, most parameters for all three heuristics showed a significant effect. We then reported results of a benchmark experiment. Our branch and bound optimal algorithm failed to find the optimal solution for our benchmark test problems, so we made use of the best upper bound found after six hours of computation. Results showed the genetic algorithm gave the best results for this group of problems, with simulated

annealing a distant third. We concluded the chapter with some observations and statistical results about the effects of computation time on solution quality and problem size on computation time. All results were as expected, except that for simulated annealing, no correlation was found between computation time and solution quality.

In the following chapter, we conduct experiments using all three heuristics for CFLP, MP-FLP, and MC-FLP on a set of larger problems. Specifically, we compare the performance of these heuristics along two dimensions of comparability: computation time allowed and solutions evaluated. We also report on the general performance without any constraints except for the parameters selected.

Chapter 8

EMPIRICAL COMPARISON RESULTS

In the last three chapters we discussed the implementation of general heuristics for solving three versions of the facilities location problem: capacitated facility location problem (CLFP), multiple-period facilities location problem (MP-FLP), and multiple commodities facilities location problem (MC-FLP). In each chapter, we then addressed the first three stages of experimentation described in our experimental design in Chapter 4 and answered the first four of seven investigative questions. In this chapter we now report the results from the fourth stage of experimentation and answer the remaining three investigative questions. In the first three sections of this chapter we report results of the comparisons of the three heuristics for each of the three FLP problem types. In the fourth section we report results of the comparisons of the three heuristics overall.

In each of the first three sections below we first describe the set of test problems use in the experiments and then report the comparative results along three dimensions which we term: "time", "information", and "unrestricted." The time dimension limits the amount of computation time each heuristic is allowed in order to arrive at its solution. All heuristics are allowed the same amount of computation time. The information dimension limits the number of candidate solutions each heuristic is allowed to evaluate such that all heuristics report a

result after the same number of candidate solutions evaluated. The unrestricted dimension allows each heuristic to report a result based on a complete run with the given parameter settings. The parameter settings for each of the heuristics are the same as those used for the benchmarking experiments. These parameter settings are used for the data collection runs along each of the three dimensions. Our original intent was to select different parameters for each of the three dimensions, but early experimentation showed this approach to be impractical for the following reasons: in the case of tabu search, a set of parameters cannot be selected for the information limited runs since the number of solutions evaluated given a set of parameters depends on the size of the FLP at hand; in the case of simulated annealing, variability in computation time makes it difficult to select a set of parameters that will result in a run length of any given time.

Given these difficulties, we chose an alternate approach. Using a single set of parameters that were identified to give good results, we adjusted the software so we could input run limitations along the time and information dimensions. The parameter settings used are the same as those used in the benchmark experiments.

8.1 CFLP Experiments

In this section we report the results of comparing the performance of the tabu search, simulated annealing, and genetic algorithm heuristics we developed in Chapter 5 for CFLP. The test problems used have larger dimensions than the test problems used for benchmark experiments. We first describe the test

problems and then provide the details of the comparisons along each of the three dimensions.

8.1.1 Test Problems

We created a new set of 20 test problems using our database of 250 U.S. cities with their inter-city distances and the 1990 census population. For a given NxM problem, the locations consist of the largest N cities, and the first M cities arranged alphabetically. The cities used in our database are listed in Appendix B. The characteristics of our test problems are shown in Table 45.

Table 45. CFLP Test Problems

Problem	Size	Capacity	Fixed Cost
CAP301	25x200	50,000	7,500
CAP302	25x200	50,000	17,500
CAP303	25x200	100,000	7,500
CAP304	25x200	100,000	17,500
CAP305	25x250	50,000	7,500
CAP306	25x250	50,000	17,500
CAP307	25x250	100,000	7,500
CAP308	25x250	100,000	17,500
CAP309	50x200	25,000	7,500
CAP310	50x200	25,000	17,500
CAP311	50x200	50,000	7,500
CAP312	50x200	50,000	17,500
CAP313	50x250	25,000	7,500
CAP314	50x250	25,000	17,500
CAP315	50x250	50,000	7,500
CAP316	50x250	50,000	17,500
CAP317	100x250	12,500	7,500
CAP318	100x250	12,500	17,500
CAP319	100x250	25,000	7,500
CAP320	100x250	25,000	17,500

For the test problems shown in Table 45, five sizes were selected: 25 locations x 200 customers, 25x250, 50x200, 50x250, and 100x250. For each problem size, four problems were generated using two levels of capacity and two levels of fixed costs. The lower capacity levels represent a capacity per potential

location such that the sum of capacities for all potential locations is approximately twice the total demand of all the customers. The higher capacity level is twice the value of the lower level. The two levels of fixed cost are $7,500 and $17,500 consistent with the fixed cost levels used by several researchers, including the original Kuehn and Hamburger data.

8.1.2 Comparative Results Along Unrestricted Dimension

As discussed in Chapter 4, we use the method of Dudewicz and Dalal (1975) as described in Law and Kelton (1982) to rank the performance of the three heuristics on each test problem. Following the recommendations from Law and Kelton, we first select a $P^* = .90$ (probability of being correct) and an $n_0 = 20$ (number of initial replications per problem for each heuristic). Following this initial set of replications, the procedure calls for selecting a d^* (indifference parameter) to determine an additional number of replications based on the variability of the initial set of replications. Due to the large computing requirements for the overall study, we reversed the procedure and computed the d^* required to make the decision based on the initial set of 20 replications. The next three tables provide a summary of the results for each heuristic. Included in these tables are the average solution value from the 20 replications, the associated standard deviation, the average time required by each heuristic for each problem, the average number of candidate solutions evaluated, and the computed d^*. Several desktop computers equipped with Pentium processors running at 120MHz were used to obtain these results.

Table 46. CFLP Tabu Search Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
CAP301	$ 542,030.85	875	770	6000000	458
CAP302	$ 756,806.10	7103	826	6000000	3720
CAP303	$ 569,522.70	3099	761	6000000	1623
CAP304	$ 692,958.85	7627	763	6000000	3994
CAP305	$1,681,249.85	2414	977	7500000	1264
CAP306	$1,921,238.30	2279	984	7500000	1193
CAP307	$1,591,298.00	0	1008	7500000	0
CAP308	$1,825,926.50	3777	979	7500000	1978
CAP309	$1,043,108.90	5838	1439	6000000	3057
CAP310	$1,454,205.50	6875	1344	6000000	3600
CAP311	$ 815,260.10	1111	1500	6000000	582
CAP312	$1,188,763.90	5272	1503	6000000	2761
CAP313	$1,282,612.70	20453	1756	7500000	10711
CAP314	$1,703,729.85	16390	1783	7500000	8583
CAP315	$ 971,058.25	2059	1881	7500000	1078
CAP316	$1,374,561.50	5923	1967	7500000	3102
CAP317	$1,165,915.50	13749	3408	7500000	7200
CAP318	$1,899,205.05	18591	3163	7500000	9736
CAP319	$ 804,823.65	2672	3602	7500000	1399
CAP320	$1,380,652.85	14141	3556	7500000	7405

Table 47. CFLP Simulated Annealing Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
CAP301	$ 548,536.50	3351	307	455117	1755
CAP302	$ 763,404.95	12631	438	485080	6615
CAP303	$ 572,060.00	4147	315	457079	2172
CAP304	$ 685,108.00	10215	408	601283	5350
CAP305	$1,681,177.00	2766	314	442987	1448
CAP306	$1,923,305.75	4719	300	438429	2471
CAP307	$1,592,800.55	1366	260	377184	715
CAP308	$1,830,591.25	3170	268	373659	1660
CAP309	$1,083,778.40	24600	446	536139	12883
CAP310	$1,510,732.10	30258	422	536406	15846
CAP311	$ 829,537.75	4685	309	362361	2454
CAP312	$1,229,490.10	17689	329	369735	9263
CAP313	$1,349,036.00	30511	471	544823	15978
CAP314	$1,784,472.45	29984	466	544974	15702
CAP315	$ 982,842.60	8448	345	384416	4424
CAP316	$1,404,225.35	14507	345	381256	7597
CAP317	$1,265,269.60	40051	755	555484	20974
CAP318	$2,090,964.05	27092	743	555420	14188
CAP319	$ 856,311.55	32914	527	409765	17236
CAP320	$1,486,803.45	26806	516	401916	14038

Table 48. CFLP Genetic Algorithm Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
CAP301	$ 680,070.85	11664	1389	84000	5949
CAP302	$ 956,609.50	16967	1638	84000	8939
CAP303	$ 709,289.70	12166	1329	84000	6034
CAP304	$ 874,224.65	12314	1332	84000	6058
CAP305	$2,763,472.10	118663	1666	86000	63724
CAP306	$3,068,314.35	114972	1631	86000	60676
CAP307	$2,537,009.70	95235	1648	86000	50784
CAP308	$2,786,853.95	101743	1635	86000	54643
CAP309	$2,204,682.40	188696	2480	84000	100483
CAP310	$2,678,116.40	155351	2489	84000	83150
CAP311	$1,529,476.90	77072	2516	84000	41189
CAP312	$1,980,293.00	84276	2518	84000	45157
CAP313	$2,930,542.75	191276	3100	86000	102749
CAP314	$3,352,771.00	171758	3086	86000	87579
CAP315	$2,018,586.75	80668	3131	86000	43336
CAP316	$2,521,051.80	122071	3128	86000	64468
CAP317	$2,920,002.60	135845	5921	86000	72890
CAP318	$3,978,389.90	163982	5929	86000	84662
CAP319	$1,897,052.85	149502	6038	85526	79518
CAP320	$2,770,106.25	138100	6038	86000	73632

Subsequently we used Friedman's F_r test for a randomized block design to test the significance of the three heuristics across all problems. Using the data in the previous three tables, we assign each heuristic a rank between one and three for each of the problems based on their average performance. However, we must first check if the difference in average performance between the heuristics is in fact significantly different. We use the computed d^* parameter to make this decision for pairwise comparisons. Namely, the average performance between tabu search and simulated annealing is significantly different if the difference between the two average values exceeds the largest of the corresponding two d^*'s. We do the same for the tabu search/genetic algorithm and simulated annealing/genetic algorithm pairs. If any pairwise comparison is not significant, then we assign each heuristic the average of the appropriate ranks. The results of these tests of significance and rankings are shown in Table 49, where the test of significance columns indicate which of the two heuristics in a pairwise comparison is best. An asterisks indicates no significant difference.

Using the rankings from Table 49, the computed Friedman statistic F_r = 35.625 (p = .0000) indicates that the performance distribution of at least two of the heuristics are in fact different from each other. To determine which distributions are different from each other, we use Wicoxon's signed rank test for paired differences. The rankings for this test are based on the absolute value of the normalized difference and are shown in Table 50. It can be seen that tabu search ranks best, followed by simulated annealing and the genetic algorithm ranks last.

Table 49. CFLP Unrestricted Runs Performance Rankings

	Test of Significance			Rankings		
Problem	**TS-SA**	**TS-GA**	**SA-GA**	**TS**	**SA**	**GA**
CAP301	TS	TS	SA	1	2	3
CAP302	*	TS	SA	1.5	1.5	3
CAP303	TS	TS	SA	1	2	3
CAP304	SA	TS	SA	2	1	3
CAP305	*	TS	SA	1.5	1.5	3
CAP306	*	TS	SA	1.5	1.5	3
CAP307	TS	TS	SA	1	2	3
CAP308	TS	TS	SA	1	2	3
CAP309	TS	TS	SA	1	2	3
CAP310	TS	TS	SA	1	2	3
CAP311	TS	TS	SA	1	2	3
CAP312	TS	TS	SA	1	2	3
CAP313	TS	TS	SA	1	2	3
CAP314	TS	TS	SA	1	2	3
CAP315	TS	TS	SA	1	2	3
CAP316	TS	TS	SA	1	2	3
CAP317	TS	TS	SA	1	2	3
CAP318	TS	TS	SA	1	2	3
CAP319	TS	TS	SA	1	2	3
CAP320	TS	TS	SA	1	2	3
		Sum of Ranks:		22.5	37.5	60
		Average Rank:		1.13	1.88	3.00

* No significant difference detected

Table 50. Wilcoxon Tests for CFLP Unrestricted Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**
-6,506	0.0120	13	-138,040	0.2547	19	-131,534	0.2398	20
0	0.0000	18	-199,803	0.2640	17	-193,205	0.2531	18
-2,537	0.0045	15	-139,767	0.2454	20	-137,230	0.2399	19
7,851	0.0113	14	-181,266	0.2616	18	-189,117	0.2760	17
0	0.0000	18	-1,082,222	0.6437	13	-1,082,295	0.6438	12
0	0.0000	18	-1,147,076	0.5971	14	-1,145,009	0.5953	14
-1,503	0.0009	17	-945,712	0.5943	15	-944,209	0.5928	15
-4,665	0.0026	16	-960,927	0.5263	16	-956,263	0.5224	16
-40,669	0.0390	7	-1,161,574	1.1136	4	-1,120,904	1.0343	5
-56,527	0.0389	8	-1,223,911	0.8416	10	-1,167,384	0.7727	11
-14,278	0.0175	11	-714,217	0.8761	9	-699,939	0.8438	9
-40,726	0.0343	9	-791,529	0.6658	12	-750,803	0.6107	13
-66,423	0.0518	5	-1,647,930	1.2848	3	-1,581,507	1.1723	3
-80,743	0.0474	6	-1,649,041	0.9679	8	-1,568,299	0.8789	7
-11,784	0.0121	12	-1,047,529	1.0787	6	-1,035,744	1.0538	4
-29,664	0.0216	10	-1,146,490	0.8341	11	-1,116,826	0.7953	10
-99,354	0.0852	2	-1,754,087	1.5045	1	-1,654,733	1.3078	1
-191,759	0.1010	1	-2,079,185	1.0948	5	-1,887,426	0.9027	6
-51,488	0.0640	4	-1,092,229	1.3571	2	-1,040,741	1.2154	2
-106,151	0.0769	3	-1,389,453	1.0064	7	-1,283,303	0.8631	8
Sum of positive Ranks:14			Sum of positive Ranks: 0			Sum of positive Ranks: 0		
Sum of negative Ranks:139			Sum of negative Ranks: 210			Sum of negative Ranks: 210		
T= 14			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 17			n= 20			n= 20		
T_0= 35			T_0= 52			T_0= 52		
Reject H_0			Reject H_0			Reject H_0		

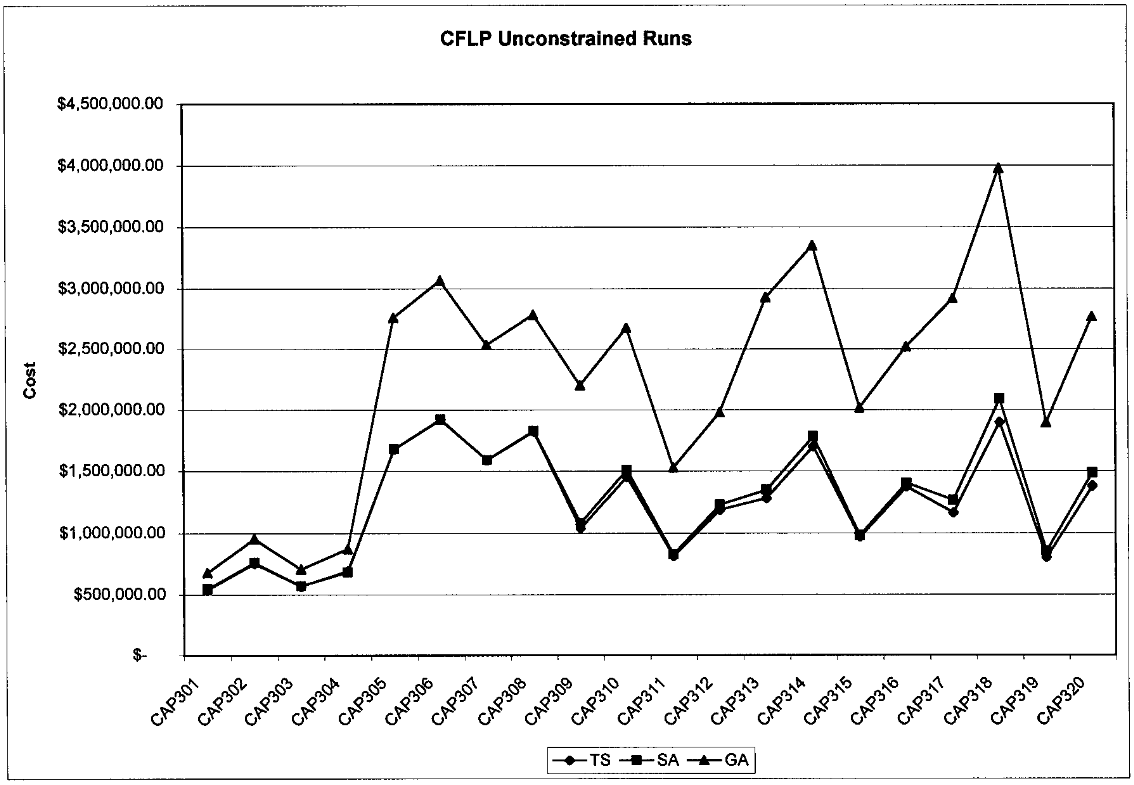

Figure 45. CFLP Unrestricted Runs Performance

Figure 45 above depicts the average performance for unrestricted runs for each heuristic. The poor performance of the genetic algorithm is obvious. The performances of tabu search and simulated annealing are quite close reflecting the lack of significant difference along some points. Nonetheless, the statistical tests have shown that tabu search does perform statistically better than simulated annealing on the average.

8.1.3 Comparative Results Along Time Constrained Dimension

For the time limited runs, we completed 20 replications of each problem using each heuristic. The same sets of parameters were used as for the unrestricted runs and the computation time was arbitrarily limited to 300 seconds (5 minutes). Results are summarized on the next three tables.

Table 51. CFLP Tabu Search Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
CAP301	$ 542,974.35	2479	300	2285000	1298
CAP302	$ 764,663.65	10769	300	2284610	5640
CAP303	$ 571,808.55	4961	300	2261630	2598
CAP304	$ 702,629.30	11588	300	2278060	6069
CAP305	$1,683,834.05	2972	300	2315988	1556
CAP306	$1,926,602.05	4112	300	2244150	2154
CAP307	$1,591,298.00	0	300	2295275	0
CAP308	$1,827,281.00	2142	300	2301425	1122
CAP309	$1,067,053.10	21589	300	1270530	11306
CAP310	$1,486,406.95	40905	300	1270920	21421
CAP311	$ 817,788.95	2197	300	1212890	1151
CAP312	$1,199,450.70	10151	300	1212210	5316
CAP313	$1,330,451.85	44227	300	1270138	23161
CAP314	$1,765,091.20	56085	300	1280563	29371
CAP315	$ 975,749.00	3995	300	1208925	2092
CAP316	$1,383,505.40	11904	300	1200700	6234
CAP317	$1,209,198.95	29604	300	641213	15503
CAP318	$1,963,164.50	32622	300	642313	17084
CAP319	$ 815,921.70	4334	300	577538	2270
CAP320	$1,438,045.90	26734	300	621500	14000

Table 52. CFLP Simulated Annealing Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
CAP301	$ 553,339.90	4311	300	338190	2258
CAP302	$ 775,511.35	14062	300	349938	7364
CAP303	$ 579,580.35	6081	300	334355	3184
CAP304	$ 701,363.00	13611	300	333155	7128
CAP305	$1,693,872.95	3583	300	340619	1876
CAP306	$1,944,007.30	7049	300	336781	3692
CAP307	$1,606,831.70	5163	300	333933	2704
CAP308	$1,843,888.80	7417	300	335681	3884
CAP309	$1,311,874.45	38522	300	280762	20173
CAP310	$1,691,904.35	39119	300	297760	20486
CAP311	$1,000,152.40	37925	300	280231	19861
CAP312	$1,468,460.60	47618	300	270427	24937
CAP313	$1,693,333.55	70267	300	274881	36798
CAP314	$2,151,809.10	54514	300	277114	28548
CAP315	$1,230,670.55	55691	300	275983	29165
CAP316	$1,699,370.40	57722	300	270276	30228
CAP317	$3,386,022.05	125912	300	210658	65939
CAP318	$4,503,614.95	138308	300	206676	72430
CAP319	$2,752,322.95	219698	300	204746	115053
CAP320	$3,365,316.05	185389	300	209684	97086

Table 53. CFLP Genetic Algorithm Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
CAP301	$ 780,834.60	21852	301	24965	11443
CAP302	$1,059,982.30	23649	300	24948	12385
CAP303	$ 811,370.65	23434	300	24950	12272
CAP304	$ 965,816.10	22930	300	24895	12008
CAP305	$3,742,680.05	215261	300	23319	112730
CAP306	$3,979,663.70	200611	301	23416	105057
CAP307	$3,409,585.35	171335	301	23315	89726
CAP308	$3,690,485.65	186350	301	23152	97589
CAP309	$3,560,788.65	170644	301	16541	89364
CAP310	$4,088,888.60	336675	301	16596	176313
CAP311	$2,521,916.15	212293	301	16696	111175
CAP312	$2,977,596.30	177897	301	16499	93163
CAP313	$5,068,200.90	310184	301	16530	162440
CAP314	$5,584,239.45	366694	301	16601	192033
CAP315	$3,325,414.00	217485	301	16574	113894
CAP316	$3,993,516.45	225865	301	16606	118283
CAP317	$8,212,998.90	444479	301	12909	232768
CAP318	$9,036,484.55	557401	301	12922	291904
CAP319	$4,712,056.15	408105	302	12856	213719
CAP320	$5,422,999.35	394072	300	12318	206370

We used Friedman's F_r to determine if the distribution of any of the heuristics is different from the others. We ranked the performance of each heuristic from one to three. If the difference in performance between any two heuristics was not statistically significant according to the $d*$ parameter, the heuristic was assigned the average of their corresponding ranks. The results of these tests of significance and the rankings are summarized in Table 54.

Using the rankings from Table 54, the computed Friedman's statistic F_r = 39.025 (p = .0000) indicates that the performance distributions of at least two of the heuristics are in fact different from each other. To determine which distributions are in fact different, we used Wilcoxon's signed rank test for paired differences. The results of these tests, shown in Table 55, indicate the performance distributions of all three heuristics are in fact different from each other. Looking at the rankings in Table 54, this is not a surprising result. The performance of tabu search is consistently better than the performance of the other two with the genetic algorithm consistently performing poorly.

Figure 46 shows that the performance of tabu search and simulated annealing is close for the smaller sized problems, but diverges for the larger sized problems. To more clearly understand the effect of limiting time, we also plotted the performance for each heuristic over time. Figure 47 shows this effect for problem CAP301. Similar charts for all problems can be found in Appendix E. Clearly, tabu search is very efficient within the first few seconds, whereas the other two heuristics are slower and the genetic algorithm levels off too early.

Table 54. CFLP Time Limited Runs Performance Rankings

	Test of Significance			Rankings		
Problem	**TS-SA**	**TS-GA**	**SA-GA**	**TS**	**SA**	**GA**
CAP301	TS	TS	SA	1	2	3
CAP302	TS	TS	SA	1	2	3
CAP303	TS	TS	SA	1	2	3
CAP304	*	TS	SA	1.5	1.5	3
CAP305	TS	TS	SA	1	2	3
CAP306	TS	TS	SA	1	2	3
CAP307	TS	TS	SA	1	2	3
CAP308	TS	TS	SA	1	2	3
CAP309	TS	TS	SA	1	2	3
CAP310	TS	TS	SA	1	2	3
CAP311	TS	TS	SA	1	2	3
CAP312	TS	TS	SA	1	2	3
CAP313	TS	TS	SA	1	2	3
CAP314	TS	TS	SA	1	2	3
CAP315	TS	TS	SA	1	2	3
CAP316	TS	TS	SA	1	2	3
CAP317	TS	TS	SA	1	2	3
CAP318	TS	TS	SA	1	2	3
CAP319	TS	TS	SA	1	2	3
CAP320	TS	TS	SA	1	2	3
			Sum of Ranks:	20.5	39.5	60
			Average Rank:	1.03	1.98	3.00

* No significant difference detected

Table 55. Wilcoxon Tests for CFLP Time Limited Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**
-10,366	0.0191	13	-237,860	0.4381	17	-227,495	0.4111	17
-10,848	0.0142	14	-295,319	0.3862	19	-284,471	0.3668	20
-7,772	0.0136	15	-239,562	0.4190	18	-231,790	0.3999	18
0	0.0000	20	-263,187	0.3746	20	-264,453	0.3771	19
-10,039	0.0060	19	-2,058,846	1.2227	13	-2,048,807	1.2095	9
-17,405	0.0090	18	-2,053,062	1.0656	15	-2,035,656	1.0471	11
-15,534	0.0098	16	-1,818,287	1.1426	14	-1,802,754	1.1219	10
-16,608	0.0091	17	-1,863,205	1.0197	16	-1,846,597	1.0015	14
-244,821	0.2294	7	-2,493,736	2.3370	7	-2,248,914	1.7143	2
-205,497	0.1383	12	-2,602,482	1.7509	11	-2,396,984	1.4167	7
-182,363	0.2230	10	-1,704,127	2.0838	9	-1,521,764	1.5215	5
-269,010	0.2243	9	-1,778,146	1.4825	12	-1,509,136	1.0277	12
-362,882	0.2728	5	-3,737,749	2.8094	4	-3,374,867	1.9930	1
-386,718	0.2191	11	-3,819,148	2.1637	8	-3,432,430	1.5951	4
-254,922	0.2613	6	-2,349,665	2.4081	6	-2,094,743	1.7021	3
-315,865	0.2283	8	-2,610,011	1.8865	10	-2,294,146	1.3500	8
-2,176,823	1.8002	2	-7,003,800	5.7921	1	-4,826,977	1.4256	6
-2,540,450	1.2941	4	-7,073,320	3.6030	3	-4,532,870	1.0065	13
-1,936,401	2.3733	1	-3,896,134	4.7751	2	-1,959,733	0.7120	15
-1,927,270	1.3402	3	-3,984,953	2.7711	5	-2,057,683	0.6114	16
Sum of positive Ranks: 0			Sum of positive Ranks: 0			Sum of positive Ranks: 0		
Sum of negative Ranks: 190			Sum of negative Ranks: 210			Sum of negative Ranks: 210		
T= 0			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 19			n= 20			n= 20		
T_0= 46			T_0= 52			T_0= 52		
Reject H_0			Reject H_0			Reject H_0		

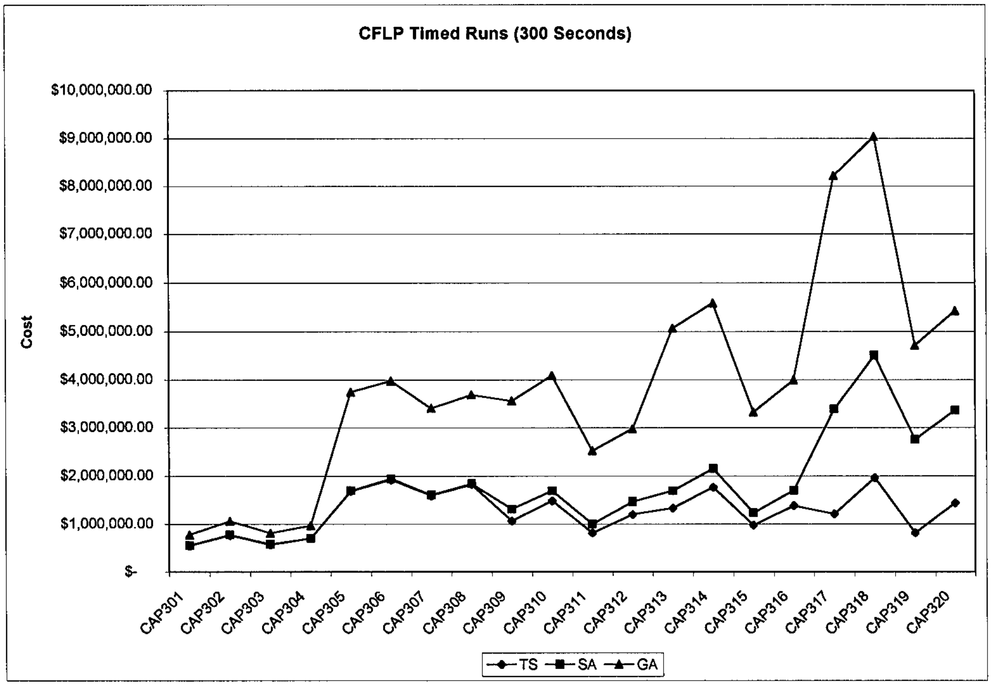

Figure 46. CFLP Time Limited Runs Performance

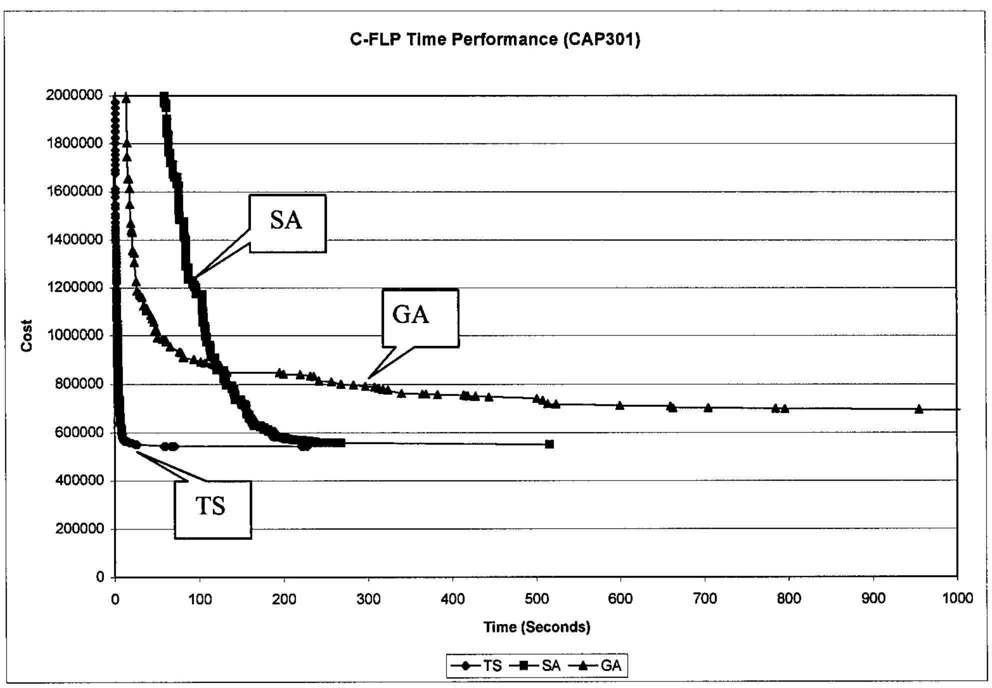

Figure 47. CFLP Heuristics Performance Over Time

8.1.4 Comparative Results Along Solutions Dimension

For the comparison along the solutions dimension, each heuristic applied was limited to a maximum number of candidate solutions it could evaluate. Of the three heuristics, the genetic algorithm evaluates the fewest solutions for the given parameters. We therefore limited tabu search and simulated annealing to the 76,040 candidate solutions the genetic algorithm evaluated for the unrestricted run. Tabu search actually evaluated a few more solutions due to the software implementation. This is because for each iteration, tabu search evaluates a number of solutions equal to the neighborhood size or a subset thereof. Since the algorithm can only be stopped between iterations, it usually exceeds the limit. In the case of simulated annealing, since the algorithm evaluates one solution at a time, it was stopped at the prescribed limit. However, simulated annealing could complete its run before reaching the prescribed limit. As before, 20 replications were completed for each problem. Parameters for tabu search and genetic algorithm were kept the same as those for the unrestricted runs. However, the parameters for simulated annealing were modified to improve its performance (RATE = .80, ACCEPT = 1000, and FACTOR = 2). The results are provided in tables 56, 57, and 58.

As before, we used the $d*$ value to test for significant differences in average performance and then ranked each heuristic using a block design. The results are shown in Table 59. The computed Friedman's F_r = 33.075 (p = .0000) indicates the performance distribution of at least one of the heuristics is different from the others. In Table 60 we show the results of using the Wilcoxon signed

rank test. This test indicates the performances of all three heuristics are significantly different from each other. Interestingly, simulated annealing turn out to be the best performer followed by tabu search and then the genetic algorithm.

Figure 48 shows the average performance of each heuristic for each of the test problems. The figure shows that even when the limit for the number of solutions allowed is drastically reduced, tabu search and simulated annealing still do much better than the genetic algorithm. However, if we look at Figure 49, we notice that for a fewer number of solutions, the genetic algorithm is superior to the other two heuristics. Its deficiency appears only after it begins to level off, whereas the other two heuristics continue to improve. If we had made the solutions cutoff smaller, then the performance of the genetic algorithm would have been relatively better. In essence, the genetic algorithm apparently is able to extract more information from a fewer number of solutions, although it also appears to reach and stay near local optima. We have included a solutions performance chart like the one in Figure 49 for each of the test problems. These can be found in Appendix F.

Table 56. CFLP Tabu Search Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
CAP301	$ 566,076.30	8587	12	76200	4497
CAP302	$ 818,214.20	14129	12	76200	7399
CAP303	$ 592,989.95	5362	12	76200	2808
CAP304	$ 751,143.00	18478	12	76200	9677
CAP305	$1,914,184.95	48849	12	76250	25581
CAP306	$2,164,626.70	48722	12	76250	25515
CAP307	$1,753,821.30	27323	12	76250	14309
CAP308	$2,003,821.30	27323	12	76250	14309
CAP309	$1,244,407.10	55873	23	76200	29260
CAP310	$1,703,818.05	71367	23	76200	37374
CAP311	$ 916,942.95	26901	23	76200	14088
CAP312	$1,356,138.50	44104	23	76200	23097
CAP313	$2,082,093.40	77865	23	76250	40777
CAP314	$2,575,829.00	93552	23	76250	48992
CAP315	$1,510,727.65	67231	23	76250	35208
CAP316	$2,005,419.90	61740	24	76250	32332
CAP317	$2,656,106.95	101685	44	76250	53251
CAP318	$3,564,357.70	124452	44	76250	65174
CAP319	$1,846,287.00	107972	45	76250	56544
CAP320	$2,605,835.70	105331	45	76250	55160

Table 57. CFLP Simulated Annealing Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
CAP301	$ 563,073.65	5063	56	76041	2652
CAP302	$ 795,292.35	15224	57	76041	7972
CAP303	$ 585,768.25	3952	58	75998	2069
CAP304	$ 725,858.65	14603	58	75986	7647
CAP305	$1,713,277.80	18402	61	75737	9637
CAP306	$1,964,911.95	19394	63	75737	10157
CAP307	$1,599,798.10	5310	61	74617	2781
CAP308	$1,843,579.20	5200	60	74821	2723
CAP309	$1,324,595.80	65872	82	76041	34497
CAP310	$1,795,437.25	74342	86	76041	38932
CAP311	$ 921,769.95	44180	79	73717	23136
CAP312	$1,358,912.30	53188	76	72388	27854
CAP313	$1,743,930.80	116671	96	76041	61099
CAP314	$2,227,589.10	106465	99	76041	55754
CAP315	$1,130,032.10	52502	87	74371	27495
CAP316	$1,583,606.35	76626	86	74048	40128
CAP317	$2,449,426.00	152632	186	76041	79931
CAP318	$3,464,948.50	152103	189	76041	79655
CAP319	$1,429,489.25	82018	147	76041	42952
CAP320	$2,181,267.75	156990	141	75328	82213

Table 58. CFLP Genetic Algorithm Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
CAP301	$ 680,070.85	11360	1392	76040	5949
CAP302	$ 956,609.50	17069	1654	76040	8939
CAP303	$ 709,289.70	11523	1328	76040	6034
CAP304	$ 874,224.65	11569	1332	76040	6058
CAP305	$2,763,472.10	121683	1666	76040	63724
CAP306	$3,068,314.35	115862	1630	76040	60676
CAP307	$2,537,009.70	96974	1647	76040	50784
CAP308	$2,786,853.95	104344	1634	76040	54643
CAP309	$2,204,682.40	191875	2478	76040	100483
CAP310	$2,678,116.40	158779	2487	76040	83150
CAP311	$1,529,476.90	78652	2515	76040	41189
CAP312	$1,980,293.00	86229	2517	76040	45157
CAP313	$2,930,542.75	196204	3098	76040	102749
CAP314	$3,352,771.00	167235	3087	76040	87579
CAP315	$2,018,586.75	82751	3130	76040	43336
CAP316	$2,521,051.80	123104	3130	76040	64468
CAP317	$2,920,002.60	139187	5919	76040	72890
CAP318	$3,978,389.90	161666	5926	76040	84662
CAP319	$1,897,052.85	151842	6040	76040	79518
CAP320	$2,770,106.25	140603	6038	76040	73632

Table 59. CFLP Solutions Limited Runs Performance Rankings

	Test of Significance			Rankings		
Problem	TS-SA	TS-GA	SA-GA	TS	SA	GA
CAP301	*	TS	SA	1.5	1.5	3
CAP302	SA	TS	SA	2	1	3
CAP303	SA	TS	SA	2	1	3
CAP304	SA	TS	SA	2	1	3
CAP305	SA	TS	SA	2	1	3
CAP306	SA	TS	SA	2	1	3
CAP307	SA	TS	SA	2	1	3
CAP308	SA	TS	SA	2	1	3
CAP309	TS	TS	SA	1	2	3
CAP310	TS	TS	SA	1	2	3
CAP311	*	TS	SA	1.5	1.5	3
CAP312	*	TS	SA	1.5	1.5	3
CAP313	SA	TS	SA	2	1	3
CAP314	SA	TS	SA	2	1	3
CAP315	SA	TS	SA	2	1	3
CAP316	SA	TS	SA	2	1	3
CAP317	SA	TS	SA	2	1	3
CAP318	SA	TS	SA	2	1	3
CAP319	SA	*	SA	2.5	1	2.5
CAP320	SA	TS	SA	2	1	3
			Sum of Ranks:	37	23.5	59.5
			Average Rank:	1.85	1.18	2.98

* No Significant difference detected

Table 60. Wilcoxon Tests for CFLP Solutions Limited Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	Absolute	Rank	Diff	Absolute	Rank	Diff	Absolute	Rank
0	0.0000	18	-113,995	0.2014	13	-116,997	0.2078	16
22,922	0.0280	15	-138,395	0.1691	15	-161,317	0.2028	18
7,222	0.0122	17	-116,300	0.1961	14	-123,521	0.2109	15
25,284	0.0337	14	-123,082	0.1639	16	-148,366	0.2044	17
200,907	0.1050	7	-849,287	0.4437	6	-1,050,194	0.6130	5
199,715	0.0923	8	-903,688	0.4175	7	-1,103,402	0.5616	8
154,023	0.0878	9	-783,188	0.4466	5	-937,212	0.5858	7
160,242	0.0800	10	-783,033	0.3908	9	-943,275	0.5117	9
-80,189	0.0644	12	-960,275	0.7717	1	-880,087	0.6644	3
-91,619	0.0538	13	-974,298	0.5718	3	-882,679	0.4916	11
0	0.0000	18	-612,534	0.6680	2	-607,707	0.6593	4
0	0.0000	18	-624,155	0.4602	4	-621,381	0.4573	12
338,163	0.1624	5	-848,449	0.4075	8	-1,186,612	0.6804	2
348,240	0.1352	6	-776,942	0.3016	11	-1,125,182	0.5051	10
380,696	0.2520	1	-507,859	0.3362	10	-888,555	0.7863	1
421,814	0.2103	3	-515,632	0.2571	12	-937,445	0.5920	6
206,681	0.0778	11	-263,896	0.0994	18	-470,577	0.1921	19
99,409	0.0279	16	-414,032	0.1162	17	-513,441	0.1482	20
416,798	0.2257	2	0	0.0000	20	-467,564	0.3271	13
424,568	0.1629	4	-164,271	0.0630	19	-588,839	0.2700	14
Sum of positive Ranks: 128			Sum of positive Ranks: 0			Sum of positive Ranks: 0		
Sum of negative Ranks: 25			Sum of negative Ranks: 190			Sum of negative Ranks: 210		
T= 25			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 17			n= 19			n= 20		
T_0= 35			T_0= 46			T_0= 52		
Reject H_0			Reject H_0			Reject H_0		

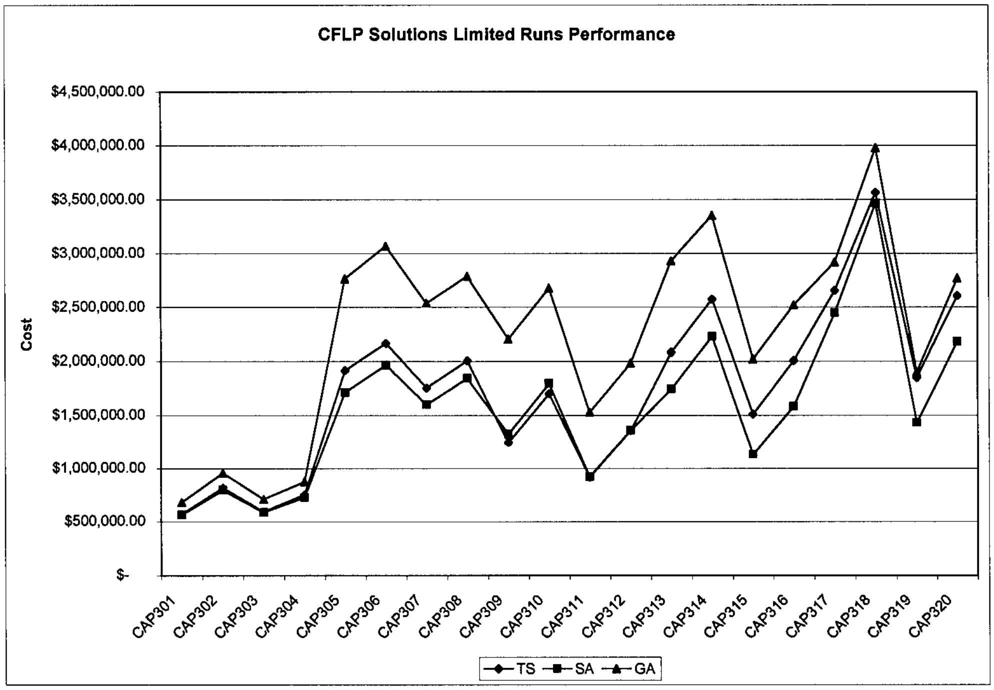

Figure 48. CFLP Solutions Limited Runs Performance

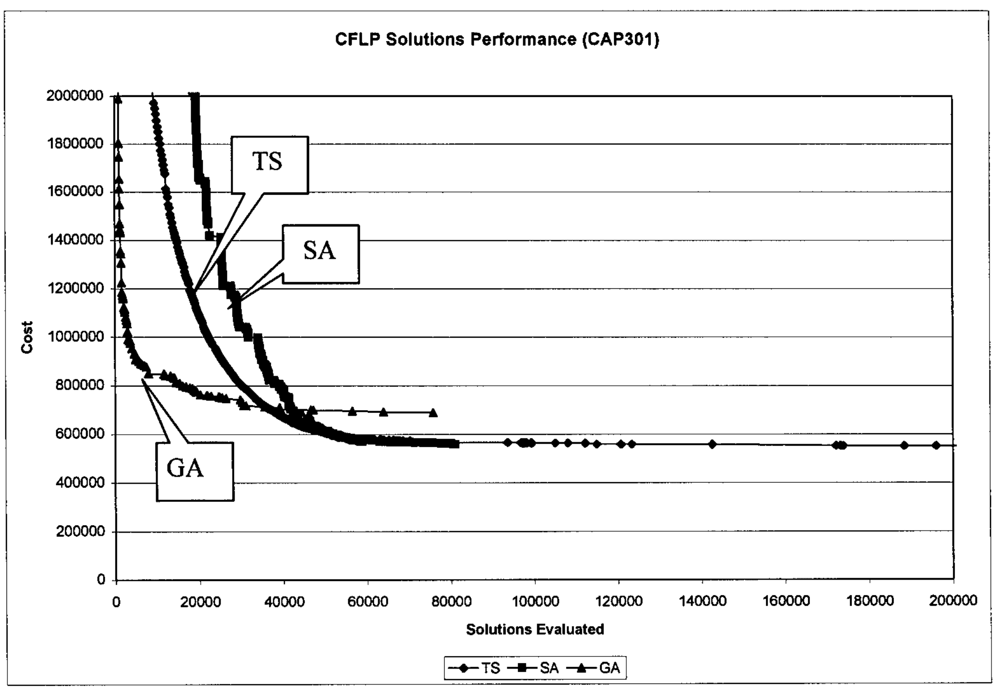

Figure 49. CFLP Heuristics Performance Over Solutions Evaluated

8.2 MP-FLP Experiments

In this section we report the results of comparing the performance of the tabu search, simulated annealing, and genetic algorithm heuristics that were discussed in Chapter 6 for MP-FLP. We used a set of larger sized test problems than our original benchmark test problems. The test problems are first described followed by the comparison results.

8.2.1 Test Problems

We created a new set of 20 test problems using our database of 250 U.S. cities. The structure of the problems is the same as was used for the benchmark problems described in §6.6 except that we increased the size dimensions. The 20 problems are described in the table below.

Table 61. MP-FLP Test Problems

Problem	Size	Capacity	Demand	Open	Close
MP301	20x15x38	2500	Increasing	4937	3702
MP302	20x15x38	2500	Increasing	2370	1777
MP303	20x15x38	5000	Increasing	4937	3702
MP304	20x15x38	5000	Increasing	2370	1777
MP305	25x10x25	2500	Decreasing	3113	2335
MP306	25x10x25	2500	Decreasing	1494	1121
MP307	25x10x25	5000	Decreasing	3113	2335
MP308	25x10x25	5000	Decreasing	1494	1121
MP309	25x15x38	5000	Concave	7646	5734
MP310	25x15x38	5000	Concave	3670	2752
MP311	25x15x38	10000	Concave	7646	5734
MP312	25x15x38	10000	Concave	3670	2752
MP313	30x10x25	2500	Convex	1660	1245
MP314	30x10x25	2500	Convex	797	598
MP315	30x10x25	5000	Convex	1660	1245
MP316	30x10x25	5000	Convex	797	598
MP317	30x15x38	2500	Seasonal	3777	2832
MP318	30x15x38	2500	Seasonal	1813	1360
MP319	30x15x38	5000	Seasonal	3777	2832
MP320	30x15x38	5000	Seasonal	1813	1360

8.2.2 Comparative Results Along Unrestricted Dimension

Following the same testing methodology described in §8.1.2, we obtained the average performance of each heuristic for each of the test problems. These results are given in Tables 62, 63, and 64. In Table 65 we summarize the test of significant differences and the resulting performance ranks for each problem in our block design. A Friedman's statistic of 39.025 ($p = .0000$) indicates the existence of some significant differences in the distributions of performance. The Wilcoxon tests, summarized in Table 66, indicate all distributions are different from each other. This is readily seen from the rankings in Table 65, which clearly shows simulated annealing turns out to be the best performer for all but one of the problems. Tabu search is a close second with the genetic algorithm lagging behind.

Figure 50 shows a graphical representation of the heuristics performance over the test problems. A visual inspection indicates that tabu search performs almost as well as simulated annealing whereas the performance of the genetic algorithm is quite distant from the other two. Also, note that the performance of the genetic algorithm worsens as the size of the problem increases. For the problems with 15 locations and 38 customers the gap between the performance of the genetic algorithm is much wider than for problems with 10 locations and 25 customers.

Table 62. MP-FLP Tabu Serach Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MP301	$ 6,324,356.75	38907	1307	11400000	20375
MP302	$ 5,153,172.15	19023	1321	11400000	9962
MP303	$ 5,472,505.55	30108	1365	11400000	15767
MP304	$ 4,402,258.45	8064	1359	11400000	4223
MP305	$ 3,745,909.80	26912	824	9375000	14094
MP306	$ 3,248,490.00	11141	799	9375000	5834
MP307	$ 3,678,300.70	23274	809	9375000	12188
MP308	$ 3,204,592.05	9242	809	9375000	4840
MP309	$12,832,359.50	66841	1678	14250000	35004
MP310	$10,636,326.70	28810	1745	14250000	15087
MP311	$11,849,066.60	39154	1713	14250000	20505
MP312	$ 9,691,844.60	16977	1709	14250000	8891
MP313	$ 2,582,253.00	11797	971	11250000	6178
MP314	$ 2,248,362.10	4869	970	11250000	2550
MP315	$ 2,567,689.35	13827	973	11250000	7241
MP316	$ 2,231,245.40	3808	980	11250000	1994
MP317	$ 7,779,605.55	24999	2025	17100000	13092
MP318	$ 6,428,051.15	14399	2027	17100000	7540
MP319	$ 7,382,018.10	23354	2062	17100000	12230
MP320	$ 6,063,820.00	12338	2074	17100000	6461

Table 63. MP-FLP Simulated Annealing Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MP301	$ 6,172,111.90	43210	507	634170	22629
MP302	$ 5,034,320.75	29396	490	623957	15394
MP303	$ 5,306,477.60	32352	313	423249	16942
MP304	$ 4,368,017.55	18345	321	430556	9607
MP305	$ 3,719,155.85	23735	289	451583	12430
MP306	$ 3,223,417.00	8505	290	469536	4454
MP307	$ 3,634,531.35	25837	252	415356	13530
MP308	$ 3,183,554.95	8343	277	440477	4369
MP309	$12,488,940.70	67691	517	594176	35449
MP310	$10,453,490.90	34117	513	593476	17867
MP311	$11,465,702.70	65494	351	426807	34299
MP312	$ 9,622,515.90	31980	352	429916	16747
MP313	$ 2,550,036.15	11196	290	438582	5863
MP314	$ 2,236,467.10	4515	300	452907	2365
MP315	$ 2,538,118.55	10426	288	442426	5460
MP316	$ 2,222,564.55	6952	294	453741	3641
MP317	$ 7,641,224.45	44264	521	520805	23181
MP318	$ 6,399,649.50	23320	506	512582	12212
MP319	$ 7,215,888.65	38824	399	436246	20332
MP320	$ 6,060,087.40	12887	396	439888	6749

Table 64. MP-FLP Genetic Algorithm Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MP301	$ 7,651,626.95	88431	1339	56030	46310
MP302	$ 6,220,335.15	82987	1335	56030	43459
MP303	$ 6,489,592.65	55207	1332	56030	28911
MP304	$ 5,129,294.60	41141	1334	56030	21545
MP305	$ 4,222,607.60	28901	761	56030	15135
MP306	$ 3,510,343.65	19648	763	56030	10290
MP307	$ 4,108,556.90	21504	760	56030	11261
MP308	$ 3,407,403.05	16312	763	56030	8542
MP309	$16,161,072.40	155468	1724	56030	81417
MP310	$13,364,279.00	179667	1692	56030	94089
MP311	$14,439,577.20	129748	1723	56030	67947
MP312	$11,630,847.20	100626	1745	56030	52697
MP313	$ 2,926,339.45	22621	916	56030	11847
MP314	$ 2,452,184.60	11919	917	56030	6242
MP315	$ 2,889,338.75	15725	907	56030	8235
MP316	$ 2,422,764.85	10429	905	56030	5462
MP317	$10,300,987.70	100285	2173	56030	52518
MP318	$ 8,620,074.95	139602	2162	56030	73108
MP319	$ 9,287,877.15	51756	2024	56030	27104
MP320	$ 7,568,532.50	65973	2028	56030	34549

Table 65. MP-FLP Unrestricted Runs Performance Rankings

	Test of Significance			Rankings		
Problem	**TS-SA**	**TS-GA**	**SA-GA**	**TS**	**SA**	**GA**
MP301	SA	TS	SA	2	1	3
MP302	SA	TS	SA	2	1	3
MP303	SA	TS	SA	2	1	3
MP304	SA	TS	SA	2	1	3
MP305	SA	TS	SA	2	1	3
MP306	SA	TS	SA	2	1	3
MP307	SA	TS	SA	2	1	3
MP308	SA	TS	SA	2	1	3
MP309	SA	TS	SA	2	1	3
MP310	SA	TS	SA	2	1	3
MP311	SA	TS	SA	2	1	3
MP312	SA	TS	SA	2	1	3
MP313	SA	TS	SA	2	1	3
MP314	SA	TS	SA	2	1	3
MP315	SA	TS	SA	2	1	3
MP316	SA	TS	SA	2	1	3
MP317	SA	TS	SA	2	1	3
MP318	SA	TS	SA	2	1	3
MP319	SA	TS	SA	2	1	3
MP320	*	TS	SA	1.5	1.5	3
			Sum of Ranks:	39.5	20.5	60
			Average Rank:	1.98	1.03	3.00

* No significant difference detected

Table 66. Wilcoxon Tests for MP-FLP Unrestricted Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**
152,245	0.0241	4	-1,327,270	0.2099	8	-1,479,515	0.2397	8
118,851	0.0231	5	-1,067,163	0.2071	9	-1,186,014	0.2356	9
166,028	0.0303	2	-1,017,087	0.1859	11	-1,183,115	0.2230	10
34,241	0.0078	12	-727,036	0.1652	12	-761,277	0.1743	12
26,754	0.0071	15	-476,698	0.1273	14	-503,452	0.1354	15
25,073	0.0077	13	-261,854	0.0806	19	-286,927	0.0890	19
43,769	0.0119	10	-430,256	0.1170	16	-474,026	0.1304	16
21,037	0.0066	16	-202,811	0.0633	20	-223,848	0.0703	20
343,419	0.0268	3	-3,328,713	0.2594	3	-3,672,132	0.2940	3
182,836	0.0172	8	-2,727,952	0.2565	5	-2,910,788	0.2785	5
383,364	0.0324	1	-2,590,511	0.2186	7	-2,973,875	0.2594	6
69,329	0.0072	14	-1,939,003	0.2001	10	-2,008,331	0.2087	11
32,217	0.0125	9	-344,086	0.1333	13	-376,303	0.1476	13
11,895	0.0053	17	-203,823	0.0907	17	-215,718	0.0965	17
29,571	0.0115	11	-321,649	0.1253	15	-351,220	0.1384	14
8,681	0.0039	19	-191,519	0.0858	18	-200,200	0.0901	18
138,381	0.0178	7	-2,521,382	0.3241	2	-2,659,763	0.3481	1
28,402	0.0044	18	-2,192,024	0.3410	1	-2,220,425	0.3470	2
166,129	0.0225	6	-1,905,859	0.2582	4	-2,071,989	0.2871	4
0	0.0000	20	-1,504,713	0.2481	6	-1,508,445	0.2489	7
Sum of positive Ranks: 190			Sum of positive Ranks: 0			Sum of positive Ranks: 0		
Sum of negative Ranks: 0			Sum of negative Ranks: 210			Sum of negative Ranks: 210		
T= 0			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 19			n= 20			n= 20		
T_0= 46			T_0= 52			T_0= 52		
Reject H_0			Reject H_0			Reject H_0		

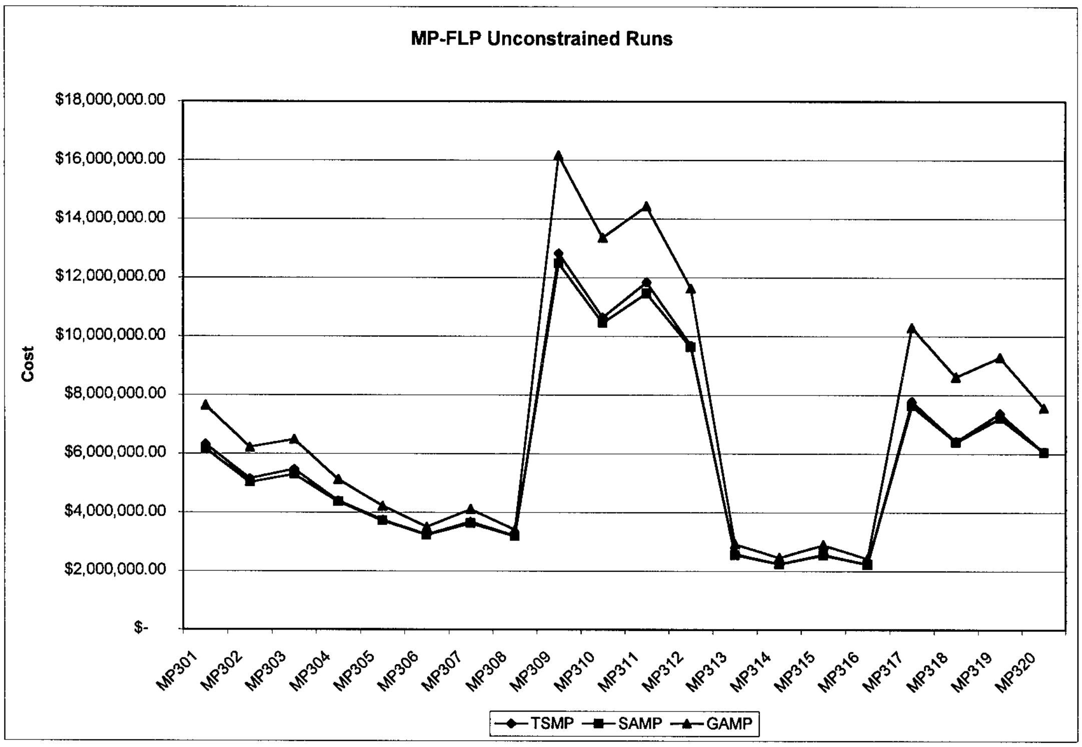

Figure 50. MP-FLP Unrestricted Runs Performance

8.2.3 Comparative Results Along Time Dimension

Consistent with the earlier experiments, we limited each replication with each heuristic to 300 seconds (5 minutes). As can be seen from the computation time, this was a considerable limitation especially for tabu search and the genetic algorithm. The results from these runs are summarized in Tables 67, 68, and 69. Table 70 provides the summarized results from testing for significant differences and the performance rankings as previously described. Note that all pairwise comparisons were found to be significant. Also note that some performance rankings have shifted from those with no time limitations. In particular, the genetic algorithm continues to be a distant third for all problems, and tabu search ranks better for several more problems than simulated annealing. The Friedman statistic of 30.4 ($p = .0000$) indicates a significant difference among the three heuristics. The Wilcoxon tests summarized in Table 71, however, indicates that only the performance distribution of the genetic algorithm is different. The performance distributions of tabu search and simulated annealing are not significantly different.

Figure 51 presents a visual representation of these results. Notice the overlaps between tabu search and simulated annealing which result in the lack of significant difference between them. Also, Figure 52 clarifies how these heuristics perform over computation time. Note that it is only at approximately 300 seconds that simulated annealing approaches the performance of tabu search. A tighter time limit would have prevented simulated annealing from being competitive. Additional plots like Figure 52 can be found in Appendix G.

Table 67. MP-FLP Tabu Serach Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MP301	$ 6,412,017.50	67895	300	2408326	35556
MP302	$ 5,183,946.00	33265	300	2389136	17420
MP303	$ 5,534,867.00	46310	300	2367362	24252
MP304	$ 4,416,786.95	18238	300	2372112	9551
MP305	$ 3,769,283.05	35590	300	3429063	18638
MP306	$ 3,258,720.90	12069	300	3224625	6321
MP307	$ 3,703,002.50	32074	300	3384375	16797
MP308	$ 3,214,193.40	12018	300	3401938	6294
MP309	$12,947,719.85	95939	300	2302610	50242
MP310	$10,684,304.00	43486	300	2293823	22773
MP311	$11,897,133.60	81998	300	2292398	42941
MP312	$ 9,737,895.60	36936	300	2286080	19343
MP313	$ 2,609,377.95	15554	300	3191025	8145
MP314	$ 2,256,592.80	10082	300	3371663	5280
MP315	$ 2,581,966.40	20916	300	3165825	10954
MP316	$ 2,238,971.05	4519	300	3371325	2366
MP317	$ 7,842,931.70	39191	300	2159958	20524
MP318	$ 6,462,526.95	20068	300	2056902	10509
MP319	$ 7,449,919.60	36820	300	1971402	19282
MP320	$ 6,087,348.30	17995	300	2169990	9424

Table 68. MP-FLP Simulated Annealing Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MP301	$ 6,917,943.35	68715	300	278181	35985
MP302	$ 5,714,941.40	61169	300	279153	32033
MP303	$ 5,665,557.20	60085	300	312706	31466
MP304	$ 4,722,668.30	47478	300	313119	24863
MP305	$ 3,719,155.85	23735	289	451583	12430
MP306	$ 3,223,417.00	8505	290	469536	4454
MP307	$ 3,634,531.35	25837	252	415356	13530
MP308	$ 3,183,554.95	8343	277	440477	4369
MP309	$14,936,867.25	166344	300	254308	87112
MP310	$12,680,310.80	98853	300	257674	51768
MP311	$12,881,783.95	158665	300	283961	83091
MP312	$10,907,744.20	132624	300	287307	69453
MP313	$ 2,550,036.15	11196	290	438582	5863
MP314	$ 2,236,467.10	4515	300	452907	2365
MP315	$ 2,538,118.55	10426	288	442426	5460
MP316	$ 2,222,564.55	6952	294	453741	3641
MP317	$10,283,323.45	112224	300	227201	58770
MP318	$ 8,774,914.15	124473	300	229071	65185
MP319	$ 8,970,455.55	112514	300	256940	58922
MP320	$ 7,550,471.70	93164	300	260682	48789

Table 69. MP-FLP Genetic Algorithm Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
MP301	$14,793,284.35	553798	300	12529	290017
MP302	$13,124,836.90	345168	300	12529	180760
MP303	$12,874,367.90	360787	300	12532	188940
MP304	$11,179,814.55	383475	300	12512	200821
MP305	$ 5,057,156.60	72103	300	21586	37759
MP306	$ 4,322,811.75	64963	300	21289	34020
MP307	$ 4,853,551.20	65462	300	21421	34282
MP308	$ 4,058,463.15	64416	300	21726	33734
MP309	$38,519,941.30	1328533	301	9696	695736
MP310	$35,192,103.75	1200063	300	9918	628458
MP311	$34,681,266.80	940257	300	9850	492400
MP312	$31,704,886.20	708200	300	9887	370875
MP313	$ 3,851,929.10	44100	300	18346	23095
MP314	$ 3,362,102.30	62417	300	18378	32687
MP315	$ 3,813,116.75	58901	300	18341	30846
MP316	$ 3,342,181.25	62909	300	18006	32945
MP317	$28,557,098.00	686920	301	8217	359731
MP318	$26,735,282.00	678667	301	8286	355409
MP319	$26,373,727.25	973598	300	8242	509861
MP320	$24,595,610.60	800262	301	8230	419087

Table 70. MP-FLP Time Limited Runs Performance Rankings

	Test of Significance			Rankings		
Problem	TS-SA	TS-GA	SA-GA	TS	SA	GA
MP301	TS	TS	SA	1	2	3
MP302	TS	TS	SA	1	2	3
MP303	TS	TS	SA	1	2	3
MP304	TS	TS	SA	1	2	3
MP305	SA	TS	SA	2	1	3
MP306	SA	TS	SA	2	1	3
MP307	SA	TS	SA	2	1	3
MP308	SA	TS	SA	2	1	3
MP309	TS	TS	SA	1	2	3
MP310	TS	TS	SA	1	2	3
MP311	TS	TS	SA	1	2	3
MP312	TS	TS	SA	1	2	3
MP313	SA	TS	SA	2	1	3
MP314	SA	TS	SA	2	1	3
MP315	SA	TS	SA	2	1	3
MP316	SA	TS	SA	2	1	3
MP317	TS	TS	SA	1	2	3
MP318	TS	TS	SA	1	2	3
MP319	TS	TS	SA	1	2	3
MP320	TS	TS	SA	1	2	3
			Sum of Ranks:	28	32	60
			Average Rank:	1.40	1.60	3.00

Table 71. Wilcoxon Tests for MP-FLP Time Limited Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**
-505,926	0.0789	10	-5,807,969	0.9058	13	-7,875,341	1.1384	12
-530,995	0.1024	8	-5,555,308	1.0716	10	-7,409,896	1.2966	10
-130,690	0.0236	12	-5,645,813	1.0200	11	-7,208,811	1.2724	11
-305,881	0.0693	11	-5,207,870	1.1791	9	-6,457,146	1.3673	9
50,127	0.0133	16	-3,325,460	0.8823	15	-1,338,001	0.3598	17
35,304	0.0108	17	-3,115,420	0.9560	12	-1,099,395	0.3411	18
68,471	0.0185	14	-2,619,415	0.7074	20	-1,219,020	0.3354	19
30,638	0.0095	18	-2,300,600	0.7158	18	-874,908	0.2748	20
-1,989,147	0.1536	6	-9,647,706	0.7451	17	-23,583,074	1.5789	8
-1,996,007	0.1868	5	-7,619,764	0.7132	19	-22,511,793	1.7753	6
-984,650	0.0828	9	-9,277,078	0.7798	16	-21,799,483	1.6923	7
-1,169,849	0.1201	7	-8,762,966	0.8999	14	-20,797,142	1.9066	4
59,342	0.0227	13	-7,266,256	2.7847	6	-1,301,893	0.5105	13
20,126	0.0089	19	-7,097,125	3.1451	4	-1,125,635	0.5033	15
43,848	0.0170	15	-6,012,301	2.3286	8	-1,274,998	0.5023	16
16,407	0.0073	20	-5,868,276	2.6210	7	-1,119,617	0.5037	14
-2,440,392	0.3112	2	-24,789,248	3.1607	3	-18,273,775	1.7770	5
-2,312,387	0.3578	1	-24,238,726	3.7507	1	-17,960,368	2.0468	2
-1,520,536	0.2041	4	-22,113,876	2.9683	5	-17,403,272	1.9401	3
-1,463,123	0.2404	3	-21,795,769	3.5805	2	-17,045,139	2.2575	1
Sum of positive Ranks: 132			Sum of positive Ranks: 0			Sum of positive Ranks: 0		
Sum of negative Ranks: 78			Sum of negative Ranks: 210			Sum of negative Ranks: 210		
T= 78			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 20			n= 20			n= 20		
T_0= 52			T_0= 52			T_0= 52		
Failed to Reject H_0			Reject H_0			Reject H_0		

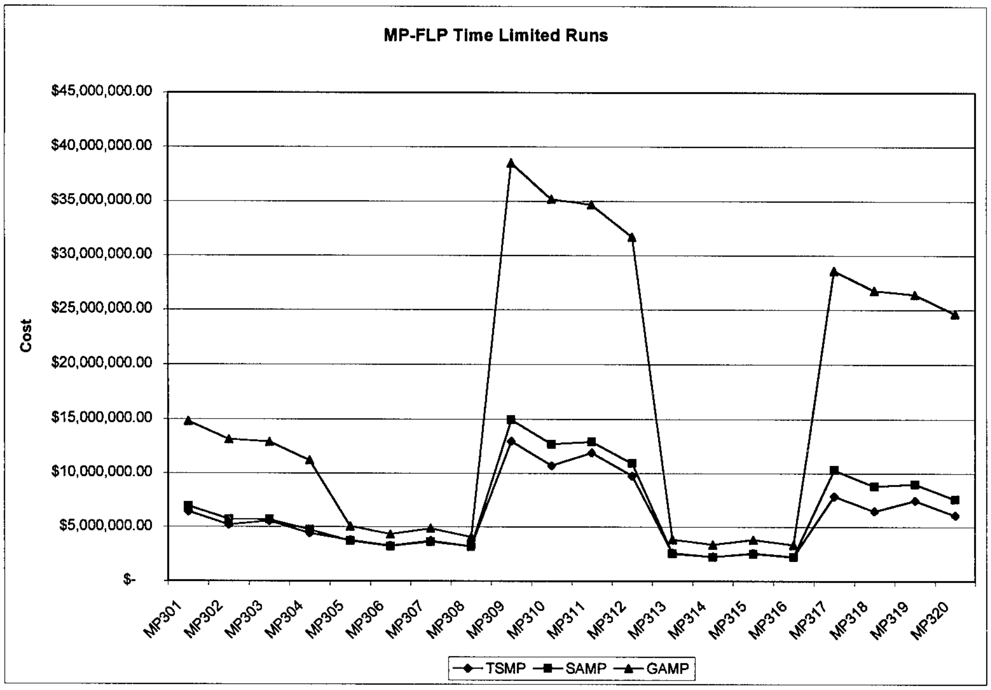

Figure 51. MP-FLP Time Limited Runs Performance

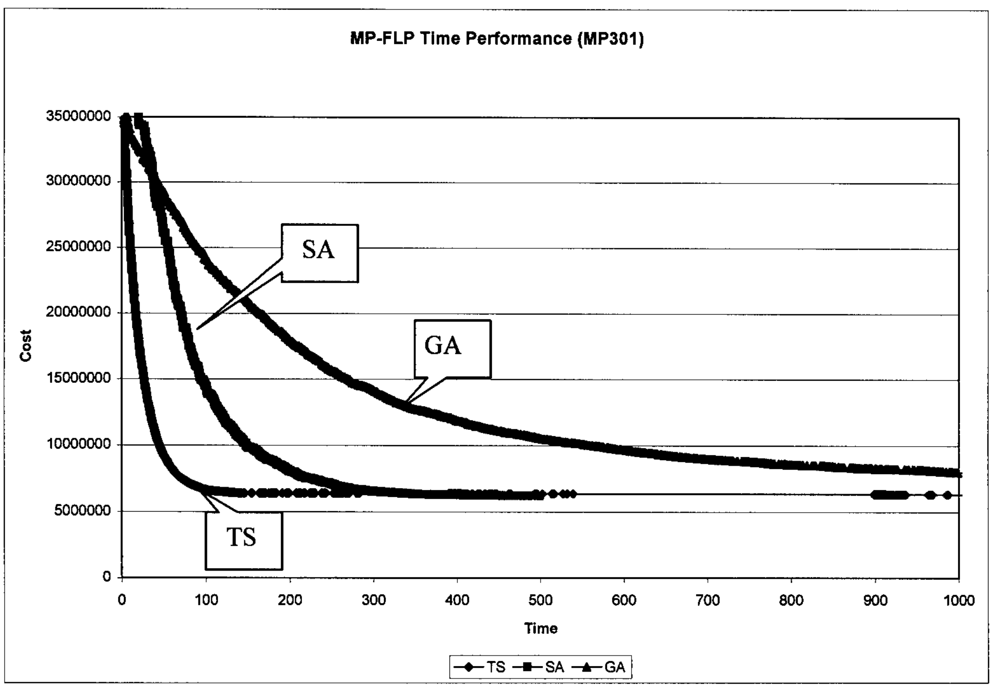

Figure 52. MP-FLP Heuristics Time Performance

8.2.4 Comparative Results Along Solutions Dimension

As was done for CFLP, we limited the number of solutions that tabu search and simulated annealing could evaluate to the number of solutions evaluated by the genetic algorithm without any restrictions (56,030 solutions). This procedure allows comparisons based on the ability of the heuristics to make the most out of the same amount of information. Tables 72, 73, and 74 summarize the average performance for each test problem for 20 replications. Table 75 summarizes the test of significance and rankings of average performance. Once again, the performance rankings have shifted. Here, the genetic algorithm is the clear winner. The Friedman statistic of 30.1 ($p = .0000$) indicates a significant difference among the three distributions. The Wilcoxon tests, summarized in Table 76, indicate the performance distribution for the genetic algorithm to be significantly different from the other two, but the distributions for tabu search and simulated annealing are not significantly different from each other.

Figure 53 illustrates how the heuristics perform. Note that for problems with more locations and customers, tabu search and simulated annealing will need to examine many more solutions to overcome the performance of the genetic algorithm. Figure 54 nicely illustrates why the genetic algorithm performs well. Basically, it extracts more information from the limited number of solutions. The other two heuristics require approximately 600,000 solutions to achieve the same results. Additional charts like the one in Figure 54 for the other test problems can be found in Appendix H.

Table 72. MP-FLP Tabu Search Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
MP301	$26,927,397.55	986101	8	56240	516408
MP302	$25,392,672.50	964549	8	56240	505122
MP303	$25,085,378.95	871869	8	56240	456586
MP304	$23,539,623.85	854635	8	56240	447561
MP305	$12,762,244.05	447365	6	56250	234279
MP306	$11,965,699.05	439772	6	56250	230303
MP307	$12,499,188.45	447425	6	56250	234311
MP308	$11,713,784.40	445937	6	56250	233532
MP309	$64,189,738.35	2129644	8	56050	1115267
MP310	$61,225,093.30	2110790	8	56050	1105393
MP311	$61,165,848.20	2609754	8	56050	1366695
MP312	$58,187,361.95	2610438	8	56050	1367053
MP313	$ 9,736,645.90	383082	6	56250	200615
MP314	$ 9,225,565.20	386212	6	56250	202254
MP315	$ 9,797,406.45	306634	6	56250	160580
MP316	$ 9,287,526.15	304738	6	56250	159587
MP317	$45,576,411.00	1436299	8	57000	752171
MP318	$43,818,993.70	1436510	9	57000	752282
MP319	$43,796,043.60	1424067	9	57000	745765
MP320	$42,034,252.25	1409747	8	57000	738266

Table 73. MP-FLP Simulated Annealing Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
MP301	$30,238,750.85	1775177	41	56031	929637
MP302	$28,814,739.15	1484589	39	56031	777460
MP303	$26,035,721.25	1292132	40	56031	676673
MP304	$24,039,100.20	1223499	41	56031	640731
MP305	$14,795,259.20	328082	35	56031	171812
MP306	$14,204,179.25	537160	34	56031	281304
MP307	$14,057,868.75	572933	34	56031	300037
MP308	$13,170,704.40	540790	35	56031	283205
MP309	$62,689,977.90	2055697	45	56031	1076542
MP310	$58,717,876.50	2735133	46	56031	1432354
MP311	$54,342,493.85	3109249	45	56031	1628274
MP312	$51,671,612.05	3283682	45	56031	1719622
MP313	$10,232,890.95	442329	37	56031	231642
MP314	$ 9,743,948.10	414338	37	56031	216984
MP315	$10,024,072.10	606997	37	56031	317876
MP316	$ 9,546,714.15	488534	37	56031	255839
MP317	$41,036,641.15	1937222	51	56031	1014498
MP318	$39,487,266.75	2077995	50	56031	1088219
MP319	$37,232,352.95	2214808	49	56031	1159867
MP320	$35,670,292.20	2216977	48	56031	1161003

Table 74. MC-FLP Genetic Algorithm Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MP301	$ 7,651,626.95	88431	1339	56030	46310
MP302	$ 6,220,335.15	82987	1335	56030	43459
MP303	$ 6,489,592.65	55207	1332	56030	28911
MP304	$ 5,129,294.60	41141	1334	56030	21545
MP305	$ 4,222,607.60	28901	761	56030	15135
MP306	$ 3,510,343.65	19648	763	56030	10290
MP307	$ 4,108,556.90	21504	760	56030	11261
MP308	$ 3,407,403.05	16312	763	56030	8542
MP309	$16,161,072.40	155468	1724	56030	81417
MP310	$13,364,279.00	179667	1692	56030	94089
MP311	$14,439,577.20	129748	1723	56030	67947
MP312	$11,630,847.20	100626	1745	56030	52697
MP313	$ 2,926,339.45	22621	916	56030	11847
MP314	$ 2,452,184.60	11919	917	56030	6242
MP315	$ 2,889,338.75	15725	907	56030	8235
MP316	$ 2,422,764.85	10429	905	56030	5462
MP317	$10,300,987.70	100285	2173	56030	52518
MP318	$ 8,620,074.95	139602	2162	56030	73108
MP319	$ 9,287,877.15	51756	2024	56030	27104
MP320	$ 7,568,532.50	65973	2028	56030	34549

Table 75. MP-FLP Solutions Limited Runs Performance Rankings

	Test of Significance			Rankings		
Problem	**TS-SA**	**TS-GA**	**SA-GA**	**TS**	**SA**	**GA**
MP301	TS	GA	GA	2	3	1
MP302	TS	GA	GA	2	3	1
MP303	TS	GA	GA	2	3	1
MP304	*	GA	GA	2.5	2.5	1
MP305	TS	GA	GA	2	3	1
MP306	TS	GA	GA	2	3	1
MP307	TS	GA	GA	2	3	1
MP308	TS	GA	GA	2	3	1
MP309	SA	GA	GA	3	2	1
MP310	SA	GA	GA	3	2	1
MP311	SA	GA	GA	3	2	1
MP312	SA	GA	GA	3	2	1
MP313	TS	GA	GA	2	3	1
MP314	TS	GA	GA	2	3	1
MP315	*	GA	GA	2.5	2.5	1
MP316	TS	GA	GA	2	3	1
MP317	SA	GA	GA	3	2	1
MP318	SA	GA	GA	3	2	1
MP319	SA	GA	GA	3	2	1
MP320	SA	GA	GA	3	2	1
			Sum of Ranks:	49	51	20
			Average Rank:	2.45	2.55	1.00

* No significant difference detected

Table 76. Wilcoxon Tests for MP-FLP Solutions Limited Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	Absolute	Rank	Diff	Absolute	Rank	Diff	Absolute	Rank
-3,311,353	0.1230	8	19,275,771	0.7158	14	22,587,124	0.7470	12
-3,422,067	0.1348	5	19,172,337	0.7550	9	22,594,404	0.7841	3
-950,342	0.0379	16	18,595,786	0.7413	11	19,546,129	0.7507	8
0	0.0000	19	18,410,329	0.7821	5	18,909,806	0.7866	2
-2,033,015	0.1593	2	8,539,636	0.6691	20	10,572,652	0.7146	17
-2,238,480	0.1871	1	8,455,355	0.7066	16	10,693,836	0.7529	7
-1,558,680	0.1247	6	8,390,632	0.6713	19	9,949,312	0.7077	20
-1,456,920	0.1244	7	8,306,381	0.7091	15	9,763,301	0.7413	15
1,499,760	0.0234	18	48,028,666	0.7482	10	46,528,906	0.7422	14
2,507,217	0.0410	15	47,860,814	0.7817	6	45,353,598	0.7724	6
6,823,354	0.1116	10	46,726,271	0.7639	8	39,902,917	0.7343	16
6,515,750	0.1120	9	46,556,515	0.8001	3	40,040,765	0.7749	5
-496,245	0.0510	14	6,810,306	0.6995	18	7,306,552	0.7140	18
-518,383	0.0562	13	6,773,381	0.7342	13	7,291,764	0.7483	11
0	0.0000	19	6,908,068	0.7051	17	7,134,733	0.7118	19
-259,188	0.0279	17	6,864,761	0.7391	12	7,123,949	0.7462	13
4,539,770	0.0996	11	35,275,423	0.7740	7	30,735,653	0.7490	10
4,331,727	0.0989	12	35,198,919	0.8033	2	30,867,192	0.7817	4
6,563,691	0.1499	4	34,508,166	0.7879	4	27,944,476	0.7505	9
6,363,960	0.1514	3	34,465,720	0.8199	1	28,101,760	0.7878	1
Sum of positive Ranks: 82			Sum of positive Ranks: 210			Sum of positive Ranks: 210		
Sum of negative Ranks: 89			Sum of negative Ranks: 0			Sum of negative Ranks: 0		
T= 82			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 18			n= 20			n= 20		
T_0= 40			T_0= 52			T_0= 52		
Failed to Reject H_0			Reject H_0			Reject H_0		

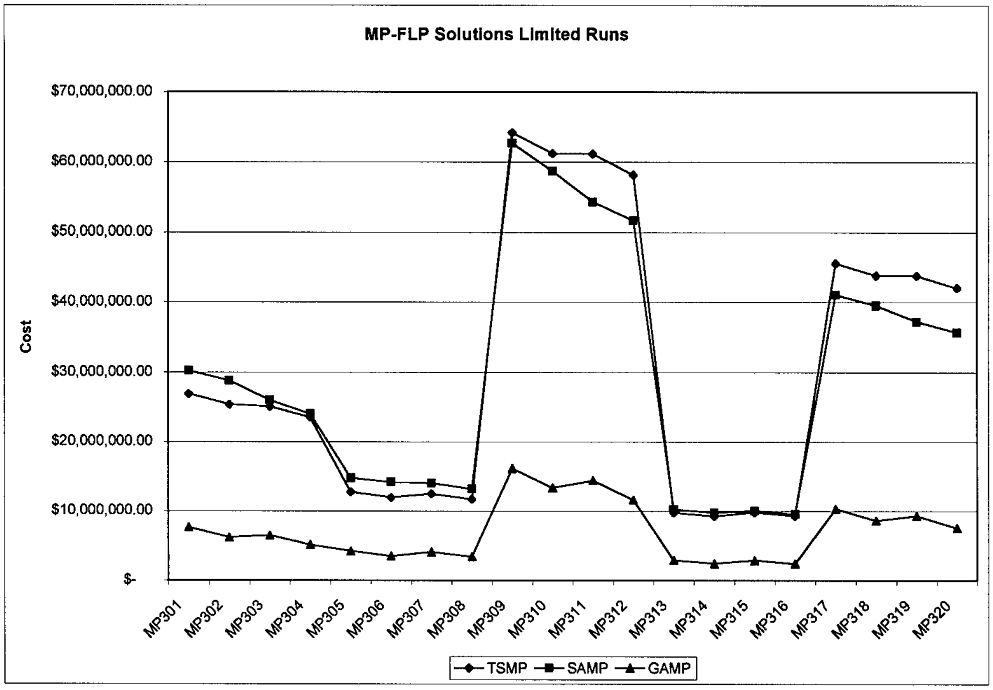

Figure 53. MP-FLP Solutions Limited Runs Performance

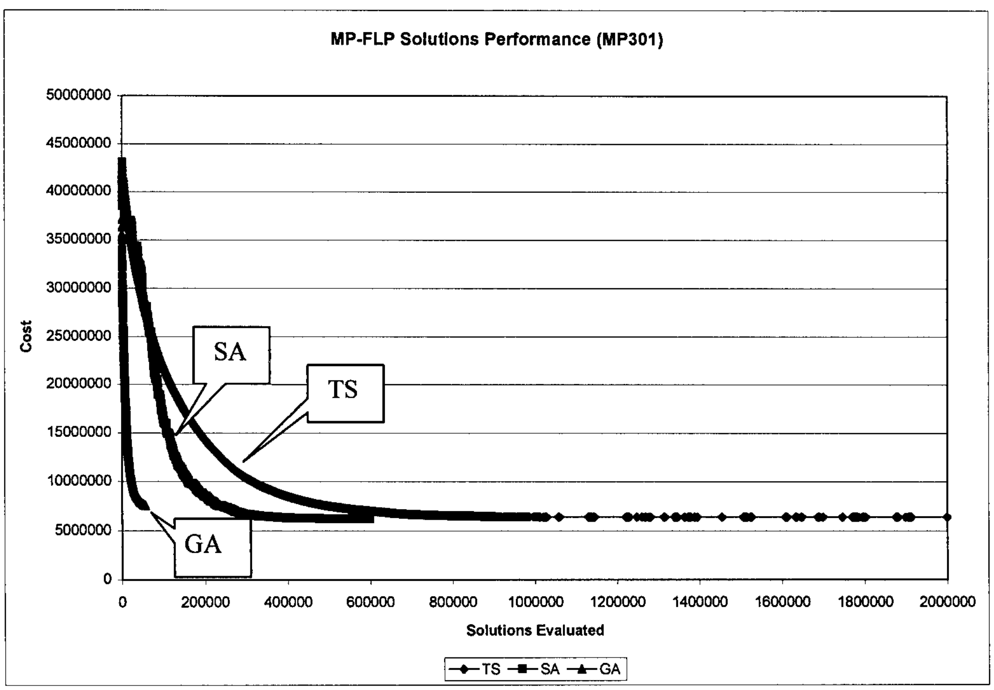

Figure 54. MP-FLP Heuristics Performance Over Solutions Evaluated

8.3 MC-FLP Experiments

In this section we report the results of comparing the performance of the tabu search, simulated annealing, and genetic algorithm heuristics we developed in Chapter 7 for MC-FLP using a set of larger test problems than our original benchmark test problems. We first describe our set of test problems and then we report our empirical comparison results along each of the three dimensions of comparability.

8.3.1 Test Problems

We created a new set of 20 test problems using our database of 250 U.S. cities (see Appendix C). The structure of the test problems is the same as for the benchmark problems in §7.6 except for their size. The set of test problems is summarized in Table 77 below.

Table 77. MC-FLP Test Problems

Problem	Size	Capacity	Fixed Costs
MC201	5x10x50	120000	90000
MC202	5x10x50	120000	180000
MC203	5x10x50	240000	90000
MC204	5x10x50	240000	180000
MC205	5x30x100	80000	90000
MC206	5x30x100	80000	180000
MC207	5x30x100	160000	90000
MC208	5x30x100	160000	180000
MC209	7x15x50	110000	90000
MC210	7x15x50	110000	180000
MC211	7x15x50	220000	90000
MC212	7x15x50	220000	180000
MC213	7x20x100	160000	90000
MC214	7x20x100	160000	180000
MC215	7x20x100	320000	90000
MC216	7x20x100	320000	180000
MC217	10x30x100	150000	90000
MC218	10x30x100	150000	180000
MC219	10x30x100	300000	90000
MC220	10x30x100	300000	180000

8.3.2 Comparative Results Along Unrestricted Dimension

For the unrestricted runs, we used the same set of parameters as was used in the benchmarking experiments. For each of the test problems, 20 replications were completed with each heuristic. The results of these runs are summarized in Tables 78, 79, and 80. Note the high variability exhibited by the simulated annealing heuristic. The results of tests of significance and rankings are shown in Table 81 and unlike the results we have seen for CFLP and MP-FLP, we did get several pairwise comparisons that were not significantly different based on the value of the d^* parameter. Fortunately, the computed Friedman statistic of 16.525 ($p = .0003$) indicates there are in fact some differences among these distributions. Surprisingly, the Wilcoxon tests (given in Table 81) indicate all of the distributions are different from each other, with tabu search on top, followed by the genetic algorithm and simulated annealing.

Figure 55 gives the average performance of the heuristics graphically for all test problems. Notice that whereas previously tabu search and simulated annealing were the two heuristics with nearly identical performance, we now have the genetic algorithm taking over the place of simulated annealing with a very strong performance. We attribute this improvement in performance to the vector representation used for MC-FLP. As we have noted, genetic algorithms appear to do well when information is limited. As the vector representation requires parameters that result in few solutions evaluated, the genetic algorithm appears to do well in this environment.

Table 78. MC-FLP Tabu Search Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MC301	$14,095,151.15	138563	57	1730	72564
MC302	$15,002,136.55	135185	56	1708	70795
MC303	$14,081,001.00	86833	64	2007	45473
MC304	$14,976,593.40	130720	63	2000	68456
MC305	$14,088,806.25	397428	3576	6570	208128
MC306	$16,175,033.00	383609	3117	6564	200891
MC307	$14,243,894.50	361380	2988	6887	189250
MC308	$16,097,081.40	381692	2474	6886	199887
MC309	$14,113,095.35	174444	135	2345	91354
MC310	$15,485,974.35	146163	133	2325	76544
MC311	$14,102,713.15	126014	175	3319	65992
MC312	$15,429,135.15	138033	174	3315	72286
MC313	$26,895,713.75	1247763	1388	3745	653438
MC314	$28,781,745.55	1226114	1363	3713	642100
MC315	$31,524,222.90	619503	1389	4620	324426
MC316	$33,164,949.90	497703	1361	4616	260641
MC317	$31,120,429.30	1047083	3382	6169	548344
MC318	$33,631,364.55	516423	3337	6134	270444
MC319	$33,076,416.35	810557	2928	6990	424478
MC320	$35,445,683.85	415686	2860	6988	217689

Table 79. MC-FLP Simulated Annealing Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MC301	$18,581,745.55	6014034	190	8302	3149472
MC302	$19,474,305.55	6007237	207	9100	3145913
MC303	$16,004,439.40	4529823	216	8536	2372210
MC304	$16,899,881.55	4523339	213	8537	2368814
MC305	$17,646,814.45	4317371	4042	7608	2260952
MC306	$19,727,446.05	4347758	3472	7095	2276865
MC307	$18,067,033.10	6779497	3385	7253	3550335
MC308	$19,937,463.10	6818670	2829	6686	3570849
MC309	$16,088,440.70	2730865	298	8377	1430119
MC310	$17,409,654.35	2726607	296	8611	1427889
MC311	$15,926,562.85	2640893	353	7473	1383002
MC312	$17,287,271.65	2600868	364	7849	1362041
MC313	$31,474,542.30	7206433	2058	7813	3773916
MC314	$33,430,733.40	7239322	2059	7885	3791140
MC315	$35,732,779.90	8310348	2554	8603	4352022
MC316	$37,427,017.15	8365056	2238	7738	4380672
MC317	$33,679,124.40	5388768	3160	7304	2822029
MC318	$36,521,290.20	5404743	3129	7278	2830394
MC319	$41,847,906.45	10486887	3797	8179	5491848
MC320	$44,340,476.95	10486143	3484	7641	5491458

Table 80. MC-FLP Genetic Algorithm Unrestricted Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MC301	$14,069,464.75	97084	196	5740	50842
MC302	$14,991,984.65	102476	194	5740	53665
MC303	$14,086,375.00	117016	184	5740	61280
MC304	$14,968,820.20	108331	188	5740	56732
MC305	$13,926,111.35	339812	3041	5740	177955
MC306	$16,040,817.60	288395	2837	5740	151029
MC307	$14,240,287.40	312442	2398	5740	163622
MC308	$16,109,400.80	197083	2125	5740	103210
MC309	$14,143,111.55	195940	333	5740	102611
MC310	$15,582,864.15	205238	319	5740	107480
MC311	$14,292,292.95	194883	292	5740	102058
MC312	$15,641,925.85	162569	286	5740	85135
MC313	$26,827,787.05	709287	2217	5740	371445
MC314	$28,187,711.95	743211	2211	5740	389210
MC315	$32,138,397.30	599932	1534	5740	314176
MC316	$33,564,295.00	505821	1518	5740	264892
MC317	$31,720,113.90	910159	3140	5740	476639
MC318	$34,882,085.70	703516	3125	5740	368422
MC319	$35,034,888.00	750397	1960	5740	392973
MC320	$37,550,354.90	665839	1951	5740	348691

Table 81. MC-FLP Unrestricted Runs Performance Rankings

	Test of Significance			Rankings		
Problem	**TS-SA**	**TS-GA**	**SA-GA**	**TS**	**SA**	**GA**
MC301	TS	*	GA	1.5	3	1.5
MC302	TS	*	GA	1.5	3	1.5
MC303	*	*	*	2	2	2
MC304	*	*	*	2	2	2
MC305	TS	*	GA	1.5	3	1.5
MC306	TS	*	GA	1.5	3	1.5
MC307	TS	*	GA	1.5	3	1.5
MC308	TS	*	GA	1.5	3	1.5
MC309	TS	*	GA	1.5	3	1.5
MC310	TS	*	GA	1.5	3	1.5
MC311	TS	TS	GA	1	3	2
MC312	TS	TS	GA	1	3	2
MC313	TS	*	GA	1.5	3	1.5
MC314	TS	*	GA	1.5	3	1.5
MC315	*	TS	*	2	2	2
MC316	*	TS	*	2	2	2
MC317	*	*	*	2	2	2
MC318	TS	TS	*	1	2.5	2.5
MC319	TS	TS	GA	1	3	2
MC320	TS	TS	GA	1	3	2
			Sum of Ranks:	30	54.5	35.5
			Average Rank:	1.50	2.73	1.78

* No significant difference detected

Table 82. Wilcoxon Tests for MC-FLP Unrestricted Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	Absolute	Rank	Diff	Absolute	Rank	Diff	Absolute	Rank
-4,486,594	0.3183	1	0	0.0000	8	4,512,281	0.2428	1
-4,472,169	0.2981	2	0	0.0000	8	4,482,321	0.2302	2
0	0.0000	16	0	0.0000	8	0	0.0000	15
0	0.0000	16	0	0.0000	8	0	0.0000	15
-3,558,008	0.2525	5	0	0.0000	8	3,720,703	0.2108	4
-3,552,413	0.2196	8	0	0.0000	8	3,686,628	0.1869	6
-3,823,139	0.2684	3	0	0.0000	8	3,826,746	0.2118	3
-3,840,382	0.2386	7	0	0.0000	8	3,828,062	0.1920	5
-1,975,345	0.1400	11	0	0.0000	8	1,945,329	0.1209	11
-1,923,680	0.1242	13	0	0.0000	8	1,826,790	0.1049	12
-1,823,850	0.1293	12	-189,580	0.0134	6	1,634,270	0.1026	13
-1,858,137	0.1204	14	-212,791	0.0138	5	1,645,346	0.0952	14
-4,578,829	0.1702	9	0	0.0000	8	4,646,755	0.1476	10
-4,648,988	0.1615	10	0	0.0000	8	5,243,021	0.1568	8
0	0.0000	16	-614,174	0.0195	4	0	0.0000	15
0	0.0000	16	-399,345	0.0120	7	0	0.0000	15
0	0.0000	16	0	0.0000	8	0	0.0000	15
-2,889,926	0.0859	15	-1,415,389	0.0421	3	0	0.0000	15
-8,771,490	0.2652	4	-1,958,472	0.0592	2	6,813,018	0.1628	7
-8,894,793	0.2509	6	-2,104,671	0.0594	1	6,790,122	0.1531	9
Sum of positive Ranks: 0			Sum of positive Ranks: 0			Sum of positive Ranks: 105		
Sum of negative Ranks: 120			Sum of negative Ranks: 28			Sum of negative Ranks: 0		
T= 0			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 15			n= 7			n= 14		
T_0= 25			T_0= 2			T_0= 21		
Reject H_0			Reject H_0			Reject H_0		

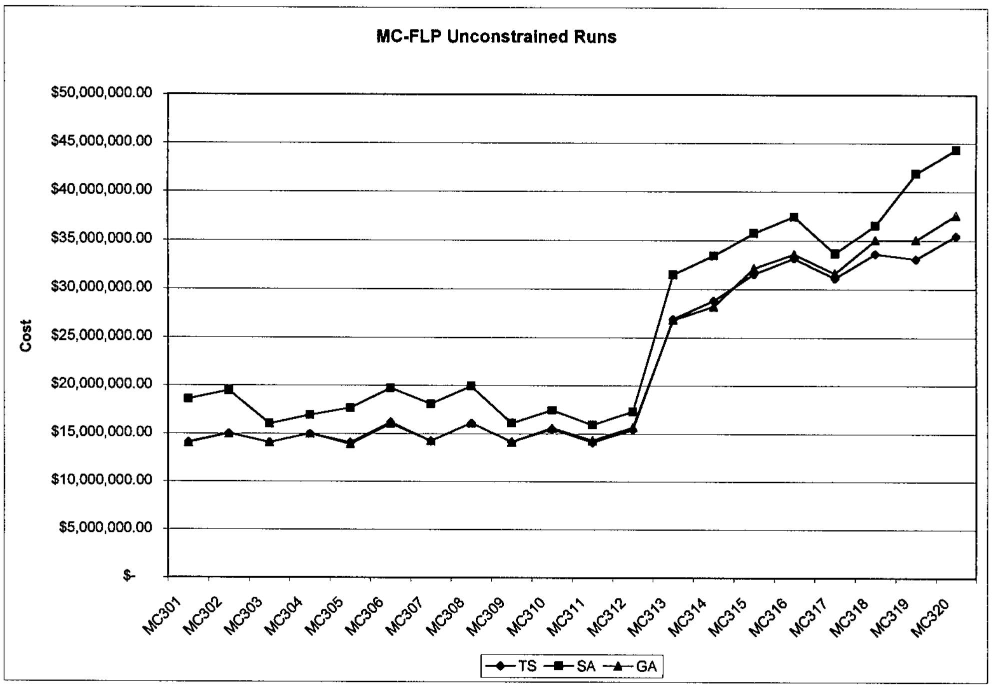

Figure 55. MC-FLP Unrestricted Runs Performance

8.3.3 Comparative Results Along Time Dimension

Tables 83, 84, and 85 summarize the results from the time limited runs for tabu search, simulated annealing, and genetic algorithm on our set of MC-FLP test problems. Each run consists of 20 replications limited to 300 seconds. Note that some of the tabu search runs finished well before this time limit for the given set of parameters. Table 86, summarizes the tests of significance and resulting rankings. Note that partly due to high variability in the results, most of the pairwise comparisons between tabu search and the genetic algorithm do not show a significant difference (the high variability results in a large d^* forcing the difference in average performances into statistical indifference). However, a Friedman statistic of 6.825 ($p = .0330$) still shows a significant difference among performance distributions. Table 87 gives the results of Wilcoxon's tests. Note that for the test between tabu search and the genetic algorithm, we do not have enough points with a significant difference to make a decision. The other two comparisons, however, show that the performance of simulated annealing is significantly poorer than for the other two heuristics.

Figure 56 is a plot of the heuristic's performances over all test problems. It clearly shows that simulated annealing does not do well, whereas tabu search and the genetic algorithm are close together at most points. Figure 57 shows some significant information. Here we notice that early on, the genetic algorithm appears to be very efficient. We see this effect for MC-FLP because the vector representation was used, which means that all three heuristics take approximately the same amount of time to evaluate each of the evaluated

solutions. Additional charts are found in Appendix I. These additional charts show that the efficiency of genetic algorithms is not universal.

Table 83. MC-FLP Tabu Search Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MC301	$14,095,151.15	138563	57	1730	72564
MC302	$15,002,136.55	135185	56	1708	70795
MC303	$14,081,001.00	86833	64	2007	45473
MC304	$14,976,593.40	130720	63	2000	68456
MC305	$17,828,469.40	4289339	306	597	2246272
MC306	$19,915,916.15	4323504	307	657	2264163
MC307	$18,159,314.30	6819341	308	774	3571201
MC308	$20,055,230.90	6821190	304	877	3572170
MC309	$14,113,095.35	174444	135	2345	91354
MC310	$15,485,974.35	146163	133	2325	76544
MC311	$14,102,713.15	126014	175	3319	65992
MC312	$15,429,135.15	138033	174	3315	72286
MC313	$30,761,558.05	5987057	302	827	3135344
MC314	$32,399,614.55	5839301	303	832	3057967
MC315	$35,075,857.00	7398693	303	1087	3874600
MC316	$36,188,725.05	6118609	303	1112	3204237
MC317	$35,001,046.75	5320748	308	588	2786407
MC318	$37,547,222.10	5224130	307	590	2735810
MC319	$42,277,496.00	10455231	307	843	5475270
MC320	$44,742,387.60	10464408	307	854	5480076

Table 84. MC-FLP Simulated Annealing Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
MC301	$18,581,745.55	6014034	190	8302	3149472
MC302	$19,474,305.55	6007237	207	9100	3145913
MC303	$16,004,439.40	4529823	216	8536	2372210
MC304	$16,899,881.55	4523339	213	8537	2368814
MC305	$18,862,834.90	4292429	300	636	2247890
MC306	$21,094,962.15	4317501	300	624	2261020
MC307	$19,126,916.30	6789131	300	679	3555381
MC308	$21,346,045.40	6856703	300	677	3590767
MC309	$16,088,440.70	2730865	298	8377	1430119
MC310	$17,409,654.35	2726607	296	8611	1427889
MC311	$15,926,562.85	2640893	353	7473	1383002
MC312	$17,287,271.65	2600868	364	7849	1362041
MC313	$35,579,711.40	7162875	300	1454	3751105
MC314	$37,367,999.40	6728212	300	1454	3523478
MC315	$39,019,980.95	8547282	300	1197	4476102
MC316	$40,379,881.60	8330661	300	1186	4362660
MC317	$41,691,240.35	5791264	300	875	3032810
MC318	$45,216,528.80	5703583	300	872	2986893
MC319	$48,715,013.05	11266213	300	758	5899971
MC320	$51,378,941.10	11032461	300	761	5777558

Table 85. MC-FLP Genetic Algorithm Time Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
MC301	$14,069,464.75	97084	196	5740	50842
MC302	$14,991,984.65	102476	194	5740	53665
MC303	$14,086,375.00	117016	184	5740	61280
MC304	$14,968,820.20	108331	188	5740	56732
MC305	$16,815,229.70	1051111	308	882	550453
MC306	$18,635,894.65	1141052	308	882	597554
MC307	$16,467,301.75	603645	306	1209	316121
MC308	$17,875,720.50	635277	307	1226	332686
MC309	$14,143,111.55	195940	333	5740	102611
MC310	$15,582,864.15	205238	319	5740	107480
MC311	$14,292,292.95	194883	292	5740	102058
MC312	$15,641,925.85	162569	286	5740	85135
MC313	$31,480,178.15	1963862	307	1034	1028449
MC314	$33,153,071.50	2361332	307	1041	1236599
MC315	$35,782,141.85	1287664	305	2033	674333
MC316	$37,604,137.00	1484578	305	2007	777454
MC317	$43,009,338.85	2488603	309	819	1303249
MC318	$46,685,740.85	3013792	308	825	1578284
MC319	$46,176,449.05	3189693	305	1870	1670401
MC320	$49,053,608.75	2935819	305	1872	1537450

Table 86. MC-FLP Time Limited Runs Performance Rankings

	Test of Significance			Rankings		
Problem	**TS-SA**	**TS-GA**	**SA-GA**	**TS**	**SA**	**GA**
MC301	TS	*	GA	2	3	1
MC302	TS	*	GA	2	3	1
MC303	*	*	*	2	2	2
MC304	*	*	*	2	2	2
MC305	*	*	*	2	2	2
MC306	*	*	GA	2.5	2.5	1
MC307	*	*	*	2	2	2
MC308	*	*	*	2	2	2
MC309	TS	*	GA	1.5	3	1.5
MC310	TS	*	GA	1.5	3	1.5
MC311	TS	TS	GA	1	3	2
MC312	TS	TS	GA	1	3	2
MC313	TS	*	GA	1.5	3	1.5
MC314	TS	*	GA	1.5	3	1.5
MC315	*	*	*	2	2	2
MC316	*	*	*	2	2	2
MC317	TS	TS	*	1	2.5	2.5
MC318	TS	TS	*	1	2.5	2.5
MC319	TS	*	*	2	2	2
MC320	TS	*	*	2	2	2
			Sum of Ranks:	34.5	49.5	36
			Average Rank:	1.73	2.48	1.80

* No significant difference detected

Table 87. Wilcoxon Tests for MC-FLP Time Limited Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	Absolute	Rank	Diff	Absolute	Rank	Diff	Absolute	Rank
-4,486,594	0.3183	1	0	0.0000	5	4,512,281	0.2428	1
-4,472,169	0.2981	2	0	0.0000	5	4,482,321	0.2302	2
0	0.0000	13	0	0.0000	5	0	0.0000	10
0	0.0000	13	0	0.0000	5	0	0.0000	10
0	0.0000	13	0	0.0000	5	0	0.0000	10
0	0.0000	13	0	0.0000	5	2,459,068	0.1166	4
0	0.0000	13	0	0.0000	5	0	0.0000	10
0	0.0000	13	0	0.0000	5	0	0.0000	10
-1,975,345	0.1400	9	0	0.0000	5	1,945,329	0.1209	3
-1,923,680	0.1242	11	0	0.0000	5	1,826,790	0.1049	7
-1,823,850	0.1293	10	-189,580	0.0134	4	1,634,270	0.1026	8
-1,858,137	0.1204	12	-212,791	0.0138	3	1,645,346	0.0952	9
-4,818,153	0.1566	5	0	0.0000	5	4,099,533	0.1152	5
-4,968,385	0.1533	6	0	0.0000	5	4,214,928	0.1128	6
0	0.0000	13	0	0.0000	5	0	0.0000	10
0	0.0000	13	0	0.0000	5	0	0.0000	10
-6,690,194	0.1911	4	-8,008,292	0.2288	2	0	0.0000	10
-7,669,307	0.2043	3	-9,138,519	0.2434	1	0	0.0000	10
-6,437,517	0.1523	7	0	0.0000	5	0	0.0000	10
-6,636,554	0.1483	8	0	0.0000	5	0	0.0000	10
Sum of positive Ranks: 0			Sum of positive Ranks: 0			Sum of positive Ranks: 45		
Sum of negative Ranks: 78			Sum of negative Ranks: 10			Sum of negative Ranks: 0		
T= 0			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 12			n= 4			n= 9		
T_0= 14			T_0= 0			T_0= 6		
Reject H_0			n is too small			Reject H_0		

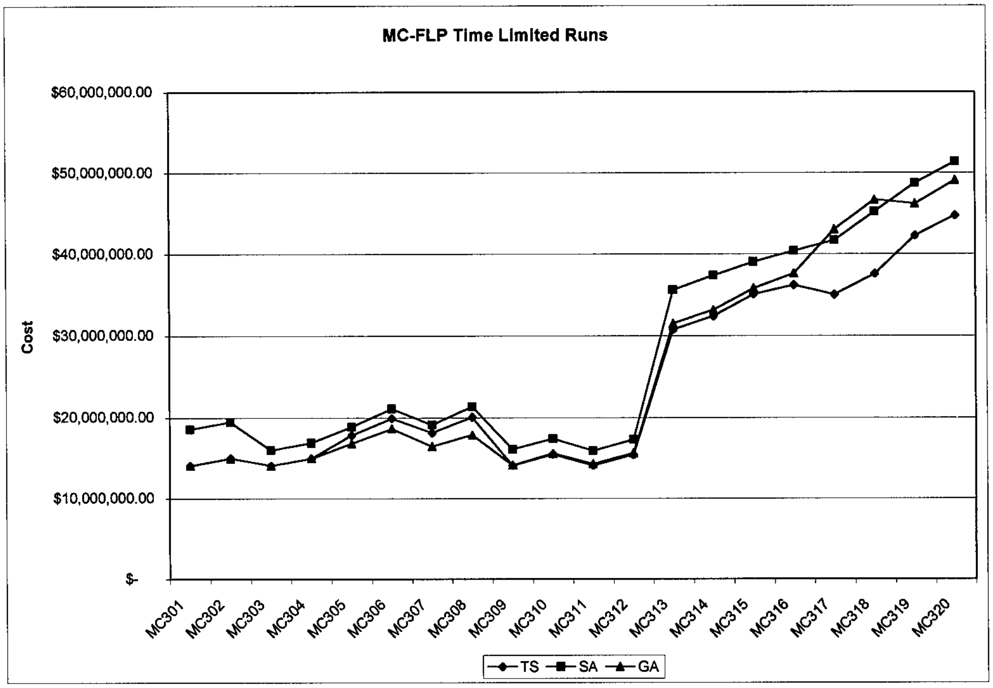

Figure 56. MC-FLP Time Limited Runs Performance

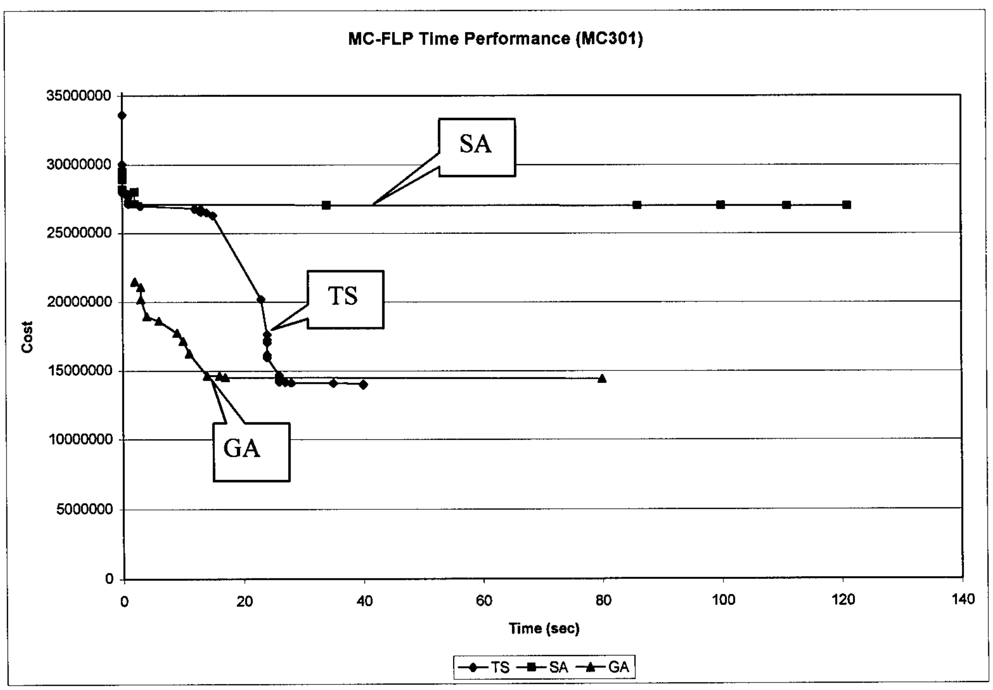

Figure 57. MC-FLP Heuristics Performance Over Time

8.3.4 Comparative Results Along Solutions Dimension

In the previous solutions limited runs, we have used the number of solutions evaluated by the genetic algorithm to limit the other two heuristics since it was the heuristic with the fewest solutions evaluated. However, for MC-FLP, the use of a vector representation means the genetic algorithm is no longer the heuristic with the fewest solutions explored. Accordingly, to determine this limit we looked at the number of solutions evaluated by each heuristic during the unrestricted runs. Then, for each group of four problems representing a specific size, we picked a lower bound. For example, for the first four problems, tabu search evaluated the fewest solutions, between 1700 and 2000. Therefore, we set the solutions limit for all the heuristics to 1500 for problems MC301-MC304. We established the limits similarly for the remaining four size groups.

The results of these runs are summarized in Tables 88, 89, and 90. The tests of significance and rankings are summarized in Table 91. A Friedman statistic of 19.075 (p = .0001) indicates there are differences among the performance distributions. The Wilcoxon tests, shown in Table 92 indicate all three heuristics perform significantly different from each other, with tabu search doing best, followed by the genetic algorithm and simulated annealing. Figure 58 illustrates the heuristics' performances over all test problems.

Figure 59 again shows how the genetic algorithm initially is much more efficient than the other two heuristics. However, tabu search is able to catch up soon enough to improve the performance over the genetic algorithm. Additional charts are found in Appendix J.

Table 88. MC-FLP Tabu Search Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	*d**
MC301	$14,110,182.45	189737	49	1504	99363
MC302	$15,007,430.05	132167	49	1503	69214
MC303	$14,104,842.35	105165	48	1504	55074
MC304	$15,011,689.65	144747	48	1504	75802
MC305	$14,096,002.35	400886	3106	5755	209939
MC306	$15,879,179.35	825511	2753	5755	432309
MC307	$14,298,066.05	415998	2469	5758	217853
MC308	$16,158,675.05	349574	2051	5756	183068
MC309	$14,132,325.75	179391	115	2006	93945
MC310	$15,522,989.45	175167	115	2005	91733
MC311	$14,207,825.85	175107	105	2006	91701
MC312	$15,494,112.20	193382	104	2006	101272
MC313	$26,955,372.70	1246575	1300	3508	652815
MC314	$28,796,617.10	1226044	1288	3508	642063
MC315	$31,755,377.90	924663	1049	3510	484234
MC316	$34,156,301.40	4019760	1031	3510	2105097
MC317	$31,170,350.00	1034045	3159	5752	541516
MC318	$33,635,539.30	518493	3129	5755	271528
MC319	$33,271,758.40	891626	2369	5753	466933
MC320	$35,642,943.45	598020	2306	5756	313175

Table 89. MC-FLP Simulated Annealing Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MC301	$18,813,931.35	5997642	29	1501	3140888
MC302	$19,741,495.30	6001728	29	1501	3143028
MC303	$16,369,109.85	4640156	31	1501	2429990
MC304	$17,220,178.70	4499524	31	1501	2356343
MC305	$17,737,324.40	4356620	3021	5741	2281506
MC306	$19,824,453.90	4331522	2878	5741	2268362
MC307	$18,100,649.20	6769090	2713	5741	3544885
MC308	$19,951,311.20	6821272	2467	5712	3572212
MC309	$16,832,137.95	2769132	64	2001	1450159
MC310	$18,035,654.25	2750340	64	2001	1440317
MC311	$16,761,036.50	2564322	78	2001	1342902
MC312	$18,130,998.20	2501231	79	2001	1309862
MC313	$32,992,887.75	7168650	767	3501	3754130
MC314	$35,078,856.30	7285360	764	3501	3815249
MC315	$36,497,576.85	8398182	951	3501	4398020
MC316	$38,229,245.55	8289005	958	3501	4340845
MC317	$33,880,960.55	5410194	2366	5741	2833249
MC318	$36,837,111.70	5354020	2354	5741	2803831
MC319	$42,169,763.60	10508766	2600	5741	5503305
MC320	$44,746,523.90	10534895	2593	5741	5516989

Table 90. MC-FLP Genetic Algorithm Solutions Limited Runs Results

Problem	Average Cost	Standard Deviation	Average Time (sec)	Avg # of Solutions	d^*
MC301	$14,354,803.80	379784	41	1522	198888
MC302	$15,324,972.70	473233	40	1522	247826
MC303	$14,673,585.95	636023	33	1522	333077
MC304	$15,412,020.80	372119	34	1522	194874
MC305	$13,926,111.35	339812	3041	5740	177955
MC306	$16,040,817.60	288395	2837	5740	151029
MC307	$14,240,287.40	312442	2398	5740	163622
MC308	$16,109,400.80	197083	2125	5740	103210
MC309	$14,630,879.75	296712	102	2016	155385
MC310	$15,944,906.80	422201	98	2016	221101
MC311	$15,357,958.65	688777	69	2016	360703
MC312	$16,840,485.15	663519	67	2016	347477
MC313	$27,354,994.20	903154	1309	3536	472970
MC314	$29,119,404.15	1065014	1304	3536	557734
MC315	$32,866,805.55	560292	762	3536	293418
MC316	$34,323,333.05	716810	769	3536	375384
MC317	$31,720,113.90	910159	3140	5740	476639
MC318	$34,882,085.70	703516	3125	5740	368422
MC319	$35,034,888.00	750397	1960	5740	392973
MC320	$37,550,354.90	665839	1951	5740	348691

Table 91. MC-FLP Solutions Limited Runs Performance Rankings

	Test of Significance			Rankings		
Problem	TS-SA	TS-GA	SA-GA	TS	SA	GA
MC301	TS	TS	GA	1	3	2
MC302	TS	TS	GA	1	3	2
MC303	*	TS	*	2	2	2
MC304	*	TS	*	2	2	2
MC305	TS	*	GA	1.5	3	1.5
MC306	TS	*	GA	1.5	3	1.5
MC307	TS	*	GA	1.5	3	1.5
MC308	TS	*	GA	1.5	3	1.5
MC309	TS	TS	GA	1	3	2
MC310	TS	TS	GA	1	3	2
MC311	TS	TS	GA	1	3	2
MC312	TS	TS	*	1	2.5	2.5
MC313	TS	*	GA	1.5	3	1.5
MC314	TS	*	GA	1.5	3	1.5
MC315	TS	TS	*	1	2.5	2.5
MC316	*	*	*	2	2	2
MC317	*	TS	*	2	2	2
MC318	TS	TS	*	1	2.5	2.5
MC319	TS	TS	GA	1	3	2
MC320	TS	TS	GA	1	3	2
			Sum of Ranks:	27	54.5	38.5
			Average Rank:	1.35	2.73	1.93

* No significant difference detected

Table 92. Wilcoxon Tests for MC-FLP Solutions Limited Runs Performance

Wilcoxon for TS-SA			Wilcoxon for TS-GA			Wilcoxon for SA-GA		
Diff	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**	**Diff**	**Absolute**	**Rank**
-4,703,749	0.3334	1	-244,621	0.0173	13	4,459,128	0.2370	1
-4,734,065	0.3154	2	-317,543	0.0212	11	4,416,523	0.2237	2
0	0.0000	17	-568,744	0.0403	5	0	0.0000	14
0	0.0000	17	-400,331	0.0267	10	0	0.0000	14
-3,641,322	0.2583	5	0	0.0000	14	3,811,213	0.2149	3
-3,945,275	0.2485	7	0	0.0000	14	3,783,636	0.1909	6
-3,802,583	0.2660	4	0	0.0000	14	3,860,362	0.2133	4
-3,792,636	0.2347	8	0	0.0000	14	3,841,910	0.1926	5
-2,699,812	0.1910	11	-498,554	0.0353	7	2,201,258	0.1308	11
-2,512,665	0.1619	14	-421,917	0.0272	9	2,090,747	0.1159	12
-2,553,211	0.1797	12	-1,150,133	0.0810	2	1,403,078	0.0837	13
-2,636,886	0.1702	13	-1,346,373	0.0869	1	0	0.0000	14
-6,037,515	0.2240	9	0	0.0000	14	5,637,894	0.1709	7
-6,282,239	0.2182	10	0	0.0000	14	5,959,452	0.1699	8
-4,742,199	0.1493	15	-1,111,428	0.0350	8	0	0.0000	14
0	0.0000	17	0	0.0000	14	0	0.0000	14
0	0.0000	17	-549,764	0.0176	12	0	0.0000	14
-3,201,572	0.0952	16	-1,246,546	0.0371	6	0	0.0000	14
-8,898,005	0.2674	3	-1,763,130	0.0530	4	7,134,876	0.1692	9
-9,103,580	0.2554	6	-1,907,411	0.0535	3	7,196,169	0.1608	10
Sum of positive Ranks: 0			Sum of positive Ranks: 0			Sum of positive Ranks: 91		
Sum of negative Ranks: 136			Sum of negative Ranks: 91			Sum of negative Ranks: 0		
T= 0			T= 0			T= 0		
α= 0.05			α= 0.05			α= 0.05		
n= 16			n= 13			n= 13		
T_0= 30			T_0= 17			T_0= 17		
Reject H_0			Reject H_0			Reject H_0		

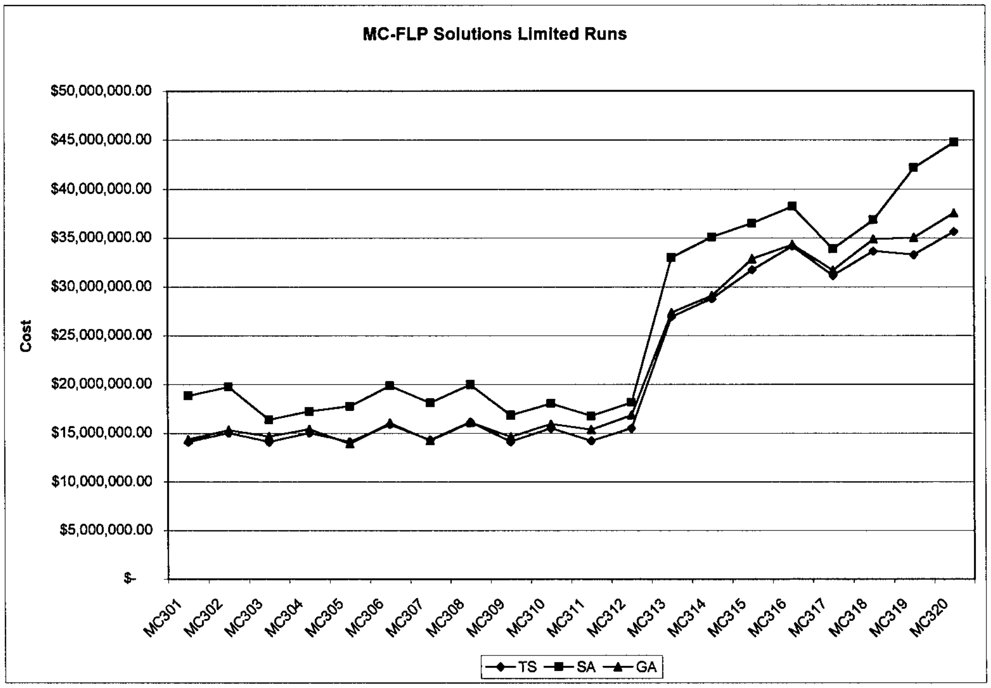

Figure 58. MC-FLP Solutions Limited Runs Performance

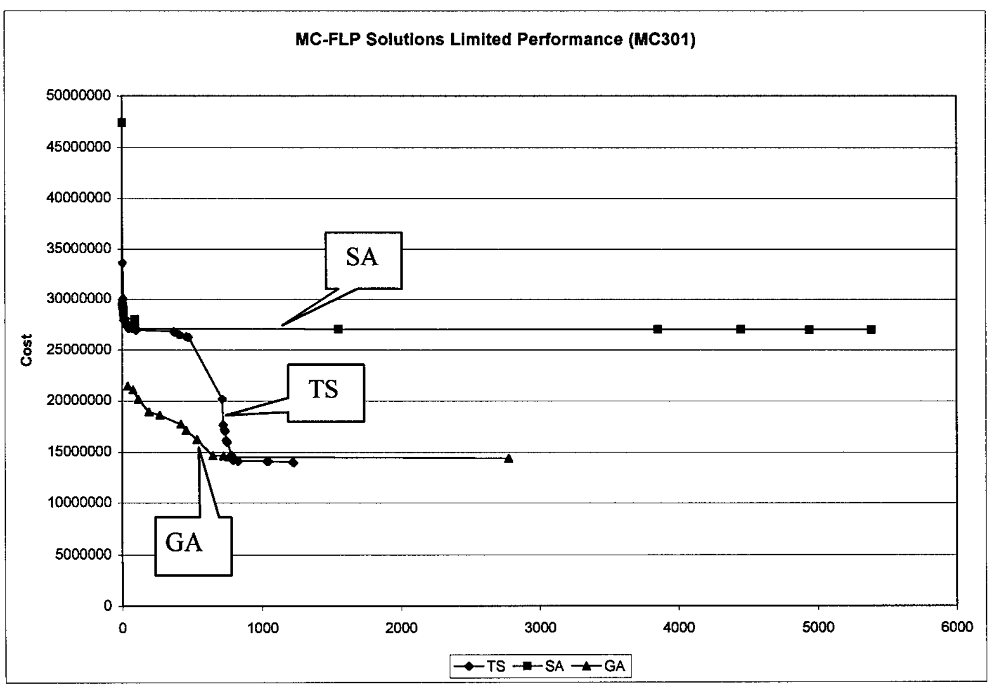

Figure 59. MC-FLP Heuristics Performance Over Solutions Evaluated

8.4 Overall Evaluation

To determine if any of the three heuristics is in fact better than the others without regard to the type of problem being solved, we pooled the results from each of the CFLP, MP-FLP, and MC-FLP experiments for each of the three comparative dimensions. For each of these dimensions, we first computed Friedman's F_r statistic to determine if there were any significant differences among the performance distributions of the three heuristics. The results in Table 93 indicate some differences for the unrestricted and time limited dimensions. However, the heuristic performance distributions for the solutions limited dimension do not show any significant difference. This is not surprising since for each problem type, a different heuristic performs best.

Table 93. Overall Comparison Results

Comparative Dimension	F_r	p
Unrestricted	35.0083	0.000000
Time Limited	44.4333	0.000000
Solutions Limited	2.2333	0.327369

To continue this analysis, we again used Wilcoxon's ranked sum test to determine which of the distributions are in fact different from each other for the two comparative dimensions with significant differences. Table 94 summarizes these results. The letter groupings indicate where significant differences exists (controlling comparisonwise error at $\alpha = .05$). These results indicate that there is not a significant difference between the two best performing heuristics, tabu search and simulated annealing, for the unrestricted dimension. However when time is limited, tabu search is significantly better than the other two heuristics.

Table 94. Overall Wilcoxon's Tests

Unrestricted Dimension			Time Dimension	
A		TS	A	TS
A	B	SA	B	SA
	B	GA	C	GA

8.5 Chapter Summary

In this chapter we presented the principal results from our empirical comparison experiments. For each of the three problem types, CFLP, MP-FLP, and MC-FLP, we reported on a new set of test problems followed by experimental results of using each of the three general heuristics on these problems. We conducted statistical tests using Friedman's F_r test for randomized block designs and Wilcoxon's signed rank test for paired differences to detect statistically significant differences in performance among and between the heuristics. Finally we conducted the same statistical tests on all results without regard to problem type. In general, the computational results showed varying performances of the heuristics depending on the problem type and the dimension of comparability.

In the next chapter we present our conclusions based on the results presented on this and previous chapters. Specifically, we will address each of the investigative questions posed in Chapter 3. We will also present suggestions for further research.

Chapter 9

CONCLUSIONS AND RECOMMENDED RESEARCH

In the previous chapter we presented the numerical results and provided a commentary on those results. In this chapter, we put all results into perspective and draw some conclusions. We proceed by first answering each of the investigative questions that were raised in this dissertation. This is followed by a summary of the principal contributions of this research. Finally, we address the limitations of this study and some ideas for future research in this area.

9.1 Answering the Investigative Questions

The data presented in the previous four chapters essentially answer in detail each of the investigative questions raised for this dissertation study. It seems appropriate that we now summarize our findings in one place and draw the conclusions. Accordingly, the following sections summarize the overall findings along each of the investigative questions.

9.1.1 Problem Representations

The first investigative question was pertaining to finding appropriate problem solution representations. For the capacitated facility location problems (CFLP), we considered three representations: a 0-1 vector, a permutation vector, and a matrix representation. Our first observation was that the permutation vector was not appropriate because of poor solutions. We found that the 0-1 vector gave the

best solution quality, but suffered from exponential growth in computation time as the size of problems grew larger. The matrix representation achieved reasonable solution quality with only a slight linear growth in computation time for larger problems. Thus, for CFLP we found the matrix representation most appropriate.

For the multiple-periods facilities location problems (MP-FLP), we again tried a 0-1 vector and matrix representations. Our results indicated that with respect to solution quality, the matrix representation was best for tabu search and the genetic algorithm whereas the 0-1 vector representation was best for simulated annealing. With respect to computation time, the matrix representation was faster for tabu search and simulated annealing. Given these results, we again found the matrix representation most appropriate for MP-FLP.

In the case of multiple-commodities facilities location problems (MC-FLP), we found that only a vector representation worked for this highly constrained problem type. The disadvantage of this kind of representation is the need to solve transportation problems to translate the solution representation into an objective function cost. Nonetheless, this was the only alternative available.

9.1.2 Parameter Effects

Once the problem solutions representations were selected, we conducted a parameter experiment so we could observe the effects of various parameter levels on solution quality and computation time. Below we summarize the findings for each of the heuristics.

For tabu search we explored the effects of three parameters: long-term memory passes (LTM), short-term memory iterations (STM), and the size of the

tabu list (TLSIZE). Our results indicate the LTM parameter has a significant impact on both solution quality and computation time for all three forms of the location problem. The STM parameter has a significant impact on solution quality for CFLP and MP-FLP, but not for MC-FLP. It does have a significant impact on computation time for all three forms of the location problem. The statistical tests showed that the TLSIZE parameter does not have a significant impact on solution quality nor computation time for any of the three versions of the location problem. Thus, for tabu search we conclude that to get the best results, both the LTM and STM parameters should be set as high as possible within the constraints of computing resources. Once these parameters are determined, an appropriate TLSIZE can be selected which works well with the other two parameters.

For simulated annealing we also explored three parameters: the cooling rate (RATE), the minimum number of solutions accepted per epoch (ACCEPT), and the maximum number of solutions evaluated per epoch expressed as a factor of ACCEPT (FACTOR). With respect to computation time, all three parameters have a significant impact in all cases, with the RATE parameter having the highest correlation between the parameter level and computation time. With respect to solution quality, all three parameters have a significant impact for CFLP and MP-FLP. For MC-FLP, none of the parameters were significant and the solution quality was consistently poor. Thus, we conclude that when simulated annealing is a useful heuristic, a parameter search is a prerequisite to gauge parameter impact on solution quality and computation time. This is the

only way to select a good set of parameters. Furthermore, since these parameters do not adjust themselves to the size of the problem, this procedure may have to be repeated for problems with significantly different size dimensions.

For genetic algorithms we explored the effects of four parameters: population size (POP), number of generations (GENS), crossover rate (XOVER), and mutation rate (MUTATE). With respect to solution quality, the two most significant factors are the number of generations and the mutation rate. The POP parameter was not significant for CFLP and MP-FLP and the XOVER parameter was not significant for any of the location problems. With respect to computation time, the POP and GENS parameters were significant as expected. The MUTATE parameter was significant for CFLP and MP-FLP, but not for MC-FLP. The XOVER parameter was not significant for any of the location problems. We conclude that selection of the MUTATE parameter is critical. The GENS parameter, which affects both solution quality and computation time, should be set as high as computing resources will allow, although higher levels of the GENS parameter achieve only marginally smaller improvements. The POP and XOVER parameters should be set after the other two are selected.

9.1.3 Benchmark Results

For CFLP and MP-FLP we were able to compare the heuristic solutions against the optimal solutions for the test problems. For MC-FLP we were not able to obtain optimal solutions to our set of benchmark problems and had to limit the comparisons against the best upper bound obtained from a branch and bound algorithm. For CFLP and MP-FLP the tabu search results had an average

gaps from the optimum of 2.32% and 6.85% respectively. Simulated annealing results had average gaps from the optimum of 3.17% and 3.36%. Genetic algorithm results were the worst as they had average gaps from the optimum of 21.28% and 15.23%. For MC-FLP, we measured the distance from the heuristic solution to the best upper bound obtained. On the average, tabu search and the genetic algorithm did better than the branch and bound upper bound with an average distance of –8.81% and –11.87% respectively. Simulated annealing did not do well with an average distance of 8.02%. From these results we conclude that tabu search and simulated annealing give reasonable results for CFLP and MP-FLP and that tabu search and genetic algorithms give reasonable results for MC-FLP.

9.1.4 Computation Time vs. Solution Quality

In general one would expect that the longer a heuristic is given to work on a certain problem, the better solutions it should obtain. Our results indicate that this is true when the heuristic is well suited (i.e., reasonable benchmark results) for the problem in the first place. Since tabu search was generally well suited for all three versions of the location problem, we found a significant correlation between time and quality in all cases. For simulated annealing, this correlation was only significant for CFLP and MP-FLP, the two versions for which it was well suited. For the genetic algorithm, the correlation was only significant for MP-FLP and CFLP, the two versions of the location problem for which it was well suited. The natural conclusion is that if the heuristic is well suited for a problem, it will achieve better results given more computation time. Conversely, heuristics not

well suited for a problem do not improve their performance given additional computation time.

9.1.5 Problem Size vs. Computation Time

As expected, our results indicate that all three heuristics require more time to solve larger sized problems (i.e., more customers, locations, periods, or commodities) for the same set of parameters. For CFLP and MP-FLP with the matrix representation, simulated annealing had the lower correlation coefficient, followed by tabu search and the genetic algorithm. Tabu search takes longer with larger problems because the size of the neighborhood it evaluates is tied to the size of the problem. The genetic algorithm takes longer with larger problems because for each solution, it must compute the resulting objective function value from scratch, which takes longer for larger problems. Tabu search and simulated annealing can compute the change in cost for each solution without having to re-compute the objective function value from scratch. Finally, the lower correlation for simulated annealing is probably due to the fact that its performance is not closely tied to the size of the problem.

For MC-FLP with the vector representation, all heuristics had a very high correlation between problem size and computation time. This is directly tied to the problem solution representation, which requires solving transportation problems.

Hence, the general conclusion is that tabu search and genetic algorithms will take more time to solve larger problems with the same set of parameters.

Furthermore, the problem representation plays a role in the magnitude of this effect.

9.1.6 Heuristics Performance for Each Problem Type

For CFLP, results show that when all heuristics were limited to the same amount of computation time (300 seconds), tabu search gave the best results. However, when all heuristics were limited to the same maximum number of candidate solutions they could evaluate, simulated annealing gave the best results. When no restrictions were placed on the heuristics, tabu search gave the best results.

For MP-FLP, results show that when all heuristics were limited to the same amount of computation time (300 seconds), tabu search and simulated annealing gave the best performance with no statistical difference between them. However, when all heuristics were limited to the same maximum number of candidate solutions they could evaluate, the genetic algorithm gave the best results. When no restrictions were placed on the heuristics simulated annealing performed best with tabu search a close second. MP-FLP presents an interesting situation because which heuristic does best depends on the basis of comparison. Of the three problem types, MP-FLP is the one that best illustrates the salient characteristics between the three heuristics: tabu search and simulated annealing are very fast, but the genetic algorithm can extract more information from fewer solutions.

For MC-FLP, results show that when all heuristics were limited to the same amount of computation time (300 seconds), tabu search and the genetic

algorithm gave the best results with no statistical difference between them. When all heuristics were limited to the same maximum number of candidate solutions evaluated, tabu search gave the best results. When the heuristics were not restricted, tabu search also gave the best results. MC-FLP is another interesting case because it is the only one where all heuristics are on an equal footing with respect to the problem solution representation (i.e., obtaining the objective function value for each solution evaluated requires the same amount of computation for all heuristics). Unlike CFLP where the genetic algorithm was not a viable heuristic, MC-FLP shows that there may be circumstances where the genetic algorithm's ability to extract much information from few solutions is the best. However, it also shows that tabu search has the same ability for some problems.

9.1.7 Overall Performance of Heuristics

Our overall results were obtained by pooling the results from the problem specific experiments. These results show that when the heuristics were given the same amount of computation time, tabu search gave the best results, followed by simulated annealing, and the genetic algorithm ranks last. When the heuristics were limited to the same maximum number of candidate solutions evaluated, the distributions of heuristic performance were not significantly different from each other. When the heuristics are not restricted, tabu search and simulated annealing performed best, with no significant difference between their performance.

9.2 Overall Conclusions

The object of this research was to determine if any of the three general heuristics studied was better than the other two. In general, we can conclude that their performance is situational principally based on two factors: problem solutions representation and measure of comparability. The problem solution representation is important because it impacts the speed at which the neighborhood-oriented heuristics can evaluate solutions. A representation which allows computing the change in cost for a "move" is very fast and favors tabu search and simulated annealing over genetic algorithms to the extent that they can evaluate more solutions in the same amount of time. However, in general it appears that genetic algorithms can extract more information from fewer solutions than the other two heuristics. Thus, the extent of the advantage tabu search and simulated annealing have depends on the differential of the number of candidate solutions they can evaluate and on the information they can extract from those solutions for a given problem type. The situation is different when the representation does not allow the neighborhood-oriented heuristics to compute the change in cost for a "move." In this case the three heuristics are in more equal footing.

The second situational factor is the measure of comparability, which has a greater impact when the problem representation favors the neighborhood-oriented heuristics. In this case, limiting computation time further limits the ability of genetic algorithms to compete with tabu search and simulated annealing. However, limiting the number of solutions evaluated, limits the ability of tabu

search and simulated annealing to compete with genetic algorithm. It should be noted that these are only general conclusions, as the experiments did not show a consistent effect throughout problem types.

Taking into consideration all of the results, we can conclude that if the problem can be represented such that a neighborhood oriented heuristic can compute the change in cost due to a "move," then tabu search will give the most robust results. If the problem cannot be represented in this manner, then tabu search and a genetic algorithm are good candidates. Furthermore, if the number of solutions to be evaluated will need to be limited, the genetic algorithm may provide a better alternative. Thus, in general we may conclude that tabu search should be tried first to the extent that it gives robust results and is easy to develop and implement.

9.3 Significant Contributions of this Study

This research has made the following significant contributions:

(1) It is the first known study to compare tabu search, simulated annealing and genetic algorithms using more than one problem type.

(2) It is the first known research to compare these algorithms that emphasizes the use of statistical methods to measure the statistical significance of their average performance.

(3) It is the first known work to consider the impact of problem solution representation on the performance of tabu search, simulated annealing, and genetic algorithms.

(4) It is the first known attempt to apply tabu search, simulated annealing, and genetic algorithms to several classical facility location problems and compare them with optimal algorithms for solution quality.

9.4 Limitations of this Study

As with any large-scale research, our conclusions must be tempered by the limitations of the method employed. In this study, the following limitations are of importance:

(1) In this study, all comparative experiments were conducted using the same set of parameters. Although these parameters were in fact selected for their good results, it is possible that other results could be obtained if the parameters were selected for each problem individually. However, this would not be practical beyond the research environment.

(2) In making comparisons, we limited both the computation time allowed and the number of solutions that each heuristic could evaluate. It is not difficult to see from our own results, that where these points are selected (amount of time, number of solutions) can significantly affect the results. The importance of our results is to clarify the general behavior of the heuristics and how their performance may be affected faced with some restriction.

(3) Although using several forms of the facilities location problem gives significant breadth to our conclusions, we must be careful not too over-generalize. By themselves, the conclusions reached in this study are limited to the problem domain we chose. However, further generalization is possible due

to our method as well as the congruence of our findings and those of preceding studies.

9.5 Future Research

What is certain is that this research adds more clarity to what we already know about the relative performance of tabu search, simulated annealing, and genetic algorithms. However, as with any large study, the findings always lead to new and significant avenues for future research.

1. Although the need for an effective problem solution representation is understood, the rationale is not well documented in the literature. This research has shown that the relative performance of these heuristics can vary depending on problem representation. However, our findings are indirect since the different representations occurred for different problems types. A focused study with the precise objective to explore the impact of solution representations may clarify how the heuristics will behave relative to each other.

2. In this research we explored the performance of the three heuristics with location problems whose size dimensions are larger than what has been published in previous location studies. However, we did not explore the impact of size on the relative performance of the heuristics. It is possible that as problems get very large, a genetic algorithm may do better than tabu search or simulated annealing due to its cutting approach. Although our current research does not seem to support this conjecture, it remains an open question.

3. On the practical side, our implementations of tabu search, simulated annealing and genetic algorithm were based on common approaches found in

the literature. However, except for tabu search, implementation is not a straightforward activity. In the case of simulated annealing, algorithm development is easy, but parameter selection is deceptively difficult. More importantly, parameters that may be good for one problem may not do well for similar problems with different dimensions. In the case of genetic algorithms, the basic algorithm is simple, but developing crossover and mutation operators can be a difficult task. Taking these observations into consideration, much more research is needed to take simulated annealing and genetic algorithm from the analysts desk to the managers toolbox, primarily because their operation and control are not intuitive. Tabu search on the other hand, can be easily implemented and explained. More importantly, the effect of the control parameters is intuitive and predictable. With this in mind, it may be time for some research that takes tabu search from the research environment to the industrial applications environment.

4. With respect to the facilities location problem, the current research can be expanded to develop a heuristic approach for solving the unified multiple-periods, multiple-commodities, capacitated facilities location problem. A natural approach for this problem would be to combine two of the heuristics studied in this research. An outer heuristic such as genetic algorithm could handle solutions to the periods and commodities decision variables while an inner neighborhood search could handle solutions to the corresponding static location problems. Similar approaches are found in the literature for other problems, but solutions to the unified facilities location problem have not been attempted.

APPENDIX A

TABLES OF REPRESENTATION SELECTION RESULTS

Table 95. TSCAP Representation Selection Results

Prob. #	Optimal Solution	TABU SEARCH FOR CFLP V Solution	V Opt Gap	P Solution	P Opt Gap	T Solution	T Opt Gap
1	$530,210.08	$530,210.08	0.00%	$559,882.62	5.60%	$550,088.63	3.75%
2	$484,258.50	$484,829.45	0.12%	$495,015.26	2.22%	$492,625.11	1.73%
3	$444,706.05	$445,260.14	0.12%	$453,052.67	1.88%	$451,244.33	1.47%
4	$393,220.20	$393,220.20	0.00%	$395,677.62	0.62%	$396,902.34	0.94%
5	$564,035.53	$564,035.52	0.00%	$582,575.65	3.29%	$584,805.37	3.68%
6	$510,583.95	$510,774.26	0.04%	$518,794.41	1.61%	$515,912.65	1.04%
7	$467,787.45	$468,341.54	0.12%	$472,178.16	0.94%	$473,205.79	1.16%
8	$416,301.60	$416,301.60	0.00%	$418,219.60	0.46%	$421,926.13	1.35%
9	$397,311.83	$398,119.76	0.20%	$444,198.53	11.80%	$433,172.83	9.03%
10	$336,284.03	$336,637.23	0.11%	$356,782.19	6.10%	$345,088.73	2.62%
11	$281,659.63	$282,616.98	0.34%	$291,204.90	3.39%	$291,329.35	3.43%
12	$205,723.05	$206,103.12	0.18%	$207,121.84	0.68%	$207,376.80	0.80%
13	$396,962.85	$399,386.67	0.61%	$429,662.59	8.24%	$414,856.74	4.51%
14	$328,687.20	$328,997.46	0.09%	$345,352.87	5.07%	$337,504.52	2.68%
15	$273,406.35	$273,808.57	0.15%	$277,960.84	1.67%	$276,789.84	1.24%
16	$199,141.70	$199,260.73	0.06%	$199,593.97	0.23%	$200,065.55	0.46%
17	$497,604.00	$497,604.00	0.00%	$555,796.02	11.69%	$553,977.66	11.33%
18	$403,928.45	$403,930.27	0.00%	$428,396.77	6.06%	$424,845.20	5.18%
19	$332,355.33	$332,355.32	0.00%	$350,815.39	5.55%	$343,544.77	3.37%
20	$251,227.63	$251,227.62	0.00%	$254,217.51	1.19%	$253,657.63	0.97%

Table 96. SACAP Representation Selection Results

Prob. #	Optimal Solution	SIMULATED ANNEALING FOR CFLP V Solution	V Opt Gap	P Solution	P Opt Gap	T Solution	T Opt Gap
1	$530,210.08	$532,331.05	0.40%	$553,411.42	4.38%	$540,571.88	1.95%
2	$484,258.50	$484,770.01	0.11%	$501,504.84	3.56%	$495,476.42	2.32%
3	$444,706.05	$444,732.93	0.01%	$455,581.93	2.45%	$452,655.43	1.79%
4	$393,220.20	$393,220.20	0.00%	$394,703.67	0.38%	$396,828.41	0.92%
5	$564,035.53	$566,258.45	0.39%	$585,654.22	3.83%	$583,146.40	3.39%
6	$510,583.95	$511,285.77	0.14%	$521,499.67	2.14%	$520,679.93	1.98%
7	$467,787.45	$469,449.72	0.36%	$474,468.45	1.43%	$478,931.01	2.38%
8	$416,301.60	$416,410.44	0.03%	$418,476.47	0.52%	$428,925.45	3.03%
9	$397,311.83	$402,318.60	1.26%	$434,781.58	9.43%	$432,073.99	8.75%
10	$336,284.03	$337,146.96	0.26%	$366,837.23	9.09%	$356,464.30	6.00%
11	$281,659.63	$282,594.21	0.33%	$295,531.62	4.93%	$291,616.68	3.54%
12	$205,723.05	$205,733.76	0.01%	$209,750.26	1.96%	$206,897.96	0.57%
13	$396,962.85	$404,864.99	1.99%	$432,567.70	8.97%	$422,950.41	6.55%
14	$328,687.20	$331,382.08	0.82%	$353,651.29	7.60%	$349,297.37	6.27%
15	$273,406.35	$275,383.94	0.72%	$283,501.69	3.69%	$275,641.00	0.82%
16	$199,141.70	$199,390.99	0.13%	$202,976.28	1.93%	$200,733.97	0.80%
17	$497,604.00	$504,815.27	1.45%	$540,961.01	8.71%	$568,871.57	14.32%
18	$403,928.45	$406,415.42	0.62%	$434,472.14	7.56%	$438,558.64	8.57%
19	$332,355.33	$333,295.37	0.28%	$355,867.39	7.07%	$356,898.76	7.38%
20	$251,227.63	$251,347.17	0.05%	$267,983.59	6.67%	$269,721.64	7.36%

Table 97. GACAP Representation Selection Results

Prob. #	Optimal Solution	GENETIC ALGORITHM FOR CFLP V Solution	V Opt Gap	P Solution	P Opt Gap	T Solution	T Opt Gap
1	$530,210.08	$533,771.61	0.67%	$702,543.16	32.50%	$633,843.51	19.55%
2	$484,258.50	$490,091.08	1.20%	$665,031.09	37.33%	$548,576.37	13.28%
3	$444,706.05	$448,209.16	0.79%	$728,295.78	63.77%	$484,101.19	8.86%
4	$393,220.20	$394,779.41	0.40%	$577,876.73	46.96%	$416,398.48	5.89%
5	$564,035.53	$575,283.35	1.99%	$783,429.03	38.90%	$658,515.37	16.75%
6	$510,583.95	$517,368.33	1.33%	$709,494.66	38.96%	$580,974.88	13.79%
7	$467,787.45	$473,162.07	1.15%	$623,755.16	33.34%	$513,990.10	9.88%
8	$416,301.60	$418,593.77	0.55%	$557,397.81	33.89%	$442,146.65	6.21%
9	$397,311.83	$415,580.43	4.60%	$750,654.86	88.93%	$582,718.59	46.67%
10	$336,284.03	$347,082.29	3.21%	$609,350.76	81.20%	$351,307.56	4.47%
11	$281,659.63	$290,500.91	3.14%	$599,287.26	112.77%	$447,373.09	58.83%
12	$205,723.05	$210,929.82	2.53%	$454,374.07	120.87%	$242,865.36	18.05%
13	$396,962.85	$417,401.18	5.15%	$837,945.84	111.09%	$564,820.35	42.29%
14	$328,687.20	$339,333.34	3.24%	$611,782.75	86.13%	$431,381.46	31.24%
15	$273,406.35	$280,778.75	2.70%	$525,028.56	92.03%	$340,528.94	24.55%
16	$199,141.70	$204,378.07	2.63%	$420,277.42	111.04%	$233,253.51	17.13%
17	$497,604.00	$515,929.64	3.68%	$852,725.50	71.37%	$658,283.49	32.29%
18	$403,928.45	$414,608.59	2.64%	$747,702.38	85.11%	$512,329.58	26.84%
19	$332,355.33	$343,059.24	3.22%	$593,851.54	78.68%	$426,144.03	28.22%
20	$251,227.63	$256,812.36	2.22%	$558,977.37	122.50%	$295,081.49	17.46%

Table 98. TSMP Representation Selection Results

		TABU SEARCH FOR MP-FLP			
	Optimal	V		T	
Problem	Solution	Solution	Opt Gap	Solution	Opt Gap
MP201	$1,364,304.00	$1,520,830.00	11.47%	$1,504,853.00	10.30%
MP202	$1,121,451.00	$1,205,967.00	7.54%	$1,216,306.00	8.46%
MP203	$1,264,892.00	$1,432,679.00	13.26%	$1,341,846.00	6.08%
MP204	$1,050,167.00	$1,179,943.00	12.36%	$1,065,173.00	1.43%
MP205	$1,077,792.00	$1,290,432.00	19.73%	$1,183,870.00	9.84%
MP206	$ 868,758.00	$ 988,762.00	13.81%	$ 890,532.00	2.51%
MP207	$1,067,015.00	$1,276,608.00	19.64%	$1,177,362.00	10.34%
MP208	$ 864,308.00	$1,045,617.00	20.98%	$ 881,095.00	1.94%
MP209	$2,294,052.00	$2,787,258.00	21.50%	$2,558,720.00	11.54%
MP210	$1,895,597.00	$2,184,731.00	15.25%	$2,021,767.00	6.66%
MP211	$2,154,639.00	$2,717,300.00	26.11%	$2,326,190.00	7.96%
MP212	$1,785,983.00	$2,285,417.00	27.96%	$1,826,457.00	2.27%
MP213	$2,447,014.00	$2,997,698.00	22.50%	$2,762,418.00	12.89%
MP214	$1,904,261.00	$2,305,392.00	21.06%	$2,118,203.00	11.23%
MP215	$2,180,598.00	$2,789,815.00	27.94%	$2,395,865.00	9.87%
MP216	$1,775,420.00	$2,304,041.00	29.77%	$1,820,100.00	2.52%
MP217	$2,506,690.00	$3,418,983.00	36.39%	$2,727,886.00	8.82%
MP218	$2,104,885.00	$2,914,903.00	38.48%	$2,188,090.00	3.95%
MP219	$2,445,718.00	$3,739,642.00	52.91%	$2,666,343.00	9.02%
MP220	$2,075,443.00	$3,385,835.00	63.14%	$2,146,739.00	3.44%

Table 99. SAMP Representation Selection Results

		SIMULATED ANNEALING FOR MP-FLP			
	Optimal	V		T	
Problem	Solution	Solution	Opt Gap	Solution	Opt Gap
MP201	$1,364,304.00	$1,456,807.00	6.78%	$1,453,614.00	6.55%
MP202	$1,121,451.00	$1,139,913.00	1.65%	$1,168,867.00	4.23%
MP203	$1,264,892.00	$1,347,241.00	6.51%	$1,305,626.00	3.22%
MP204	$1,050,167.00	$1,061,882.00	1.12%	$1,065,273.00	1.44%
MP205	$1,077,792.00	$1,108,773.00	2.87%	$1,183,879.00	9.84%
MP206	$ 868,758.00	$ 882,398.00	1.57%	$ 887,064.00	2.11%
MP207	$1,067,015.00	$1,146,729.00	7.47%	$1,165,215.00	9.20%
MP208	$ 864,308.00	$ 882,326.00	2.08%	$ 880,544.00	1.88%
MP209	$2,294,052.00	$2,422,286.00	5.59%	$2,532,350.00	10.39%
MP210	$1,895,597.00	$1,921,796.00	1.38%	$2,020,787.00	6.60%
MP211	$2,154,639.00	$2,249,995.00	4.43%	$2,305,354.00	6.99%
MP212	$1,785,983.00	$1,839,017.00	2.97%	$1,841,307.00	3.10%
MP213	$2,447,014.00	$2,570,473.00	5.05%	$2,794,558.00	14.20%
MP214	$1,904,261.00	$1,954,981.00	2.66%	$2,104,446.00	10.51%
MP215	$2,180,598.00	$2,288,135.00	4.93%	$2,409,272.00	10.49%
MP216	$1,775,420.00	$1,820,694.00	2.55%	$1,825,663.00	2.83%
MP217	$2,506,690.00	$2,623,874.00	4.67%	$2,670,119.00	6.52%
MP218	$2,104,885.00	$2,157,559.00	2.50%	$2,185,748.00	3.84%
MP219	$2,445,718.00	$2,576,975.00	5.37%	$2,612,418.00	6.82%
MP220	$2,075,443.00	$2,116,860.00	2.00%	$2,136,009.00	2.92%

Table 100. GAMP Representation Selection Results

		GENETIC ALGORITHM FOR MP-FLP			
	Optimal	V		T	
Problem	Solution	Solution	Opt Gap	Solution	Opt Gap
MP201	$1,364,304.00	$1,541,517.00	12.99%	$1,498,688.00	9.85%
MP202	$1,121,451.00	$1,215,905.00	8.42%	$1,158,197.00	3.28%
MP203	$1,264,892.00	$1,469,712.00	16.19%	$1,358,722.00	7.42%
MP204	$1,050,167.00	$1,187,904.00	13.12%	$1,053,315.00	0.30%
MP205	$1,077,792.00	$1,289,167.00	19.61%	$1,409,935.00	30.82%
MP206	$ 868,758.00	$ 972,387.00	11.93%	$1,064,059.00	22.48%
MP207	$1,067,015.00	$1,287,760.00	20.69%	$1,315,040.00	23.24%
MP208	$ 864,308.00	$1,035,655.00	19.82%	$ 985,746.00	14.05%
MP209	$2,294,052.00	$2,631,467.00	14.71%	$2,522,621.00	9.96%
MP210	$1,895,597.00	$2,106,340.00	11.12%	$1,912,630.00	0.90%
MP211	$2,154,639.00	$2,591,323.00	20.27%	$2,354,664.00	9.28%
MP212	$1,785,983.00	$2,114,844.00	18.41%	$1,796,661.00	0.60%
MP213	$2,447,014.00	$2,871,571.00	17.35%	$2,689,198.00	9.90%
MP214	$1,904,261.00	$2,218,480.00	16.50%	$1,982,294.00	4.10%
MP215	$2,180,598.00	$2,737,183.00	25.52%	$2,503,094.00	14.79%
MP216	$1,775,420.00	$2,136,103.00	20.32%	$1,804,090.00	1.61%
MP217	$2,506,690.00	$3,065,915.00	22.31%	$3,287,554.00	31.15%
MP218	$2,104,885.00	$2,545,313.00	20.92%	$2,619,108.00	24.43%
MP219	$2,445,718.00	$3,089,294.00	26.31%	$3,157,236.00	29.09%
MP220	$2,075,443.00	$2,539,420.00	22.36%	$2,522,234.00	21.53%

APPENDIX B

TABLES OF PARAMETER EXPERIMENTATION RESULTS

Table 101. Tabu Search Parameter Experiment Results (CFLP)

LTM	STM	TLSIZE	Average Cost	Variance of Cost	Average Time (sec)	Variance of Time	Average Solutions
1	1000	5	$580,607.24	634736294	8.8	0.12	50000
1	1000	7	$588,960.88	573699825	8.6	0.06	50000
1	1000	12	$579,384.75	447593634	8.7	0.05	50000
1	1500	5	$579,574.54	656852604	12.9	0.15	75000
1	1500	7	$588,048.86	582916071	12.6	0.11	75000
1	1500	12	$576,937.09	491492763	12.6	0.12	75000
1	2000	5	$578,657.53	678149548	16.6	0.17	100000
1	2000	7	$585,896.01	605668905	16.6	0.19	100000
1	2000	12	$574,868.06	482965879	16.6	0.24	100000
1	2500	5	$572,483.01	517523459	20.5	0.30	125000
1	2500	7	$584,294.11	668244700	20.6	0.33	125000
1	2500	12	$574,023.82	471781921	20.4	0.37	125000
1	3000	5	$572,342.81	518650256	25.1	0.55	150000
1	3000	7	$584,086.59	670095080	24.4	0.85	150000
1	3000	12	$572,224.24	466441558	24.4	0.93	150000
5	1000	5	$553,297.61	305661594	41.3	0.63	250000
5	1000	7	$550,002.04	117333569	41.4	0.13	250000
5	1000	12	$554,167.12	242141784	41.8	0.19	250000
5	1500	5	$551,576.62	168218373	61.7	0.93	375000
5	1500	7	$547,652.96	70458178	62.3	4.22	375000
5	1500	12	$549,203.95	201225103	61.0	2.04	375000
5	2000	5	$560,597.58	272048257	81.9	1.55	500000
5	2000	7	$551,387.80	194075971	82.4	1.04	500000
5	2000	12	$546,373.43	297487233	79.9	2.16	500000
5	2500	5	$549,043.54	178318288	102.2	5.64	625000
5	2500	7	$549,699.81	216568260	101.5	5.56	625000
5	2500	12	$554,591.28	300308257	100.0	3.83	625000
5	3000	5	$546,291.78	201368154	123.4	2.69	750000
5	3000	7	$543,332.89	92475995	121.2	4.51	750000
5	3000	12	$542,460.52	282033474	121.3	5.76	750000
10	1000	5	$543,735.14	98072051	82.4	1.86	500000
10	1000	7	$540,161.05	97118788	84.0	1.40	500000
10	1000	12	$544,786.88	94129687	81.6	1.45	500000
10	1500	5	$545,097.53	138572493	123.8	3.65	750000
10	1500	7	$542,856.64	60607071	122.3	3.53	750000
10	1500	12	$541,714.26	215812028	121.5	1.67	750000
10	2000	5	$550,600.07	61805530	162.7	6.55	1000000
10	2000	7	$549,014.51	141295961	162.4	7.95	1000000
10	2000	12	$539,089.27	87577231	160.9	3.92	1000000
10	2500	5	$540,948.72	225418390	202.9	6.77	1250000
10	2500	7	$541,896.60	101570616	203.2	9.93	1250000
10	2500	12	$536,073.76	46378153	199.0	6.11	1250000
10	**3000**	**5**	**$533,817.99**	**142882533**	**246.6**	**25.17**	**1500000**
10	3000	7	$537,844.85	61050630	247.6	46.93	1500000
10	3000	12	$534,139.20	72358892	243.9	40.14	1500000

Table 102. Simulated Annealing Parameter Experiment Results (CFLP)

RATE	ACCEPT	FACTOR	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
0.75	1000	1	$584,715.10	438819600	32.8	15	35200
0.75	1000	2	$574,347.73	155701061	51.6	61	56255
0.75	1000	3	$579,427.61	65475672	70.8	20	78421
0.75	1500	1	$569,442.47	64411950	49.5	20	54900
0.75	1500	2	$574,270.69	320663836	82.1	43	92350
0.75	1500	3	$561,489.96	289547496	98.0	195	110966
0.75	2000	1	$569,118.75	245762858	63.7	24	71200
0.75	2000	2	$567,621.90	134823565	106.5	166	121055
0.75	2000	3	$563,359.70	172605278	134.7	419	153285
0.75	2500	1	$565,015.06	324102337	86.3	11	96000
0.75	2500	2	$564,369.34	322837936	132.9	88	151202
0.75	2500	3	$559,840.24	192295669	168.2	542	193881
0.80	1000	1	$572,916.32	265230412	39.1	18	43200
0.80	1000	2	$566,103.07	384883056	67.7	75	75869
0.80	1000	3	$563,249.52	196666076	88.0	136	99647
0.80	1500	1	$584,906.76	376524118	60.3	31	67650
0.80	1500	2	$558,312.31	87045251	107.5	80	108904
0.80	1500	3	$564,940.68	313906304	133.7	149	152557
0.80	2000	1	$573,217.91	266189472	82.2	30	93000
0.80	2000	2	$563,604.66	149010303	127.8	225	146212
0.80	2000	3	$558,483.42	410217537	174.8	249	200368
0.80	2500	1	$567,669.10	336079275	103.7	100	118500
0.80	2500	2	$553,520.73	117391829	163.0	249	188194
0.80	2500	3	$553,500.13	201485705	209.2	707	241599
0.85	1000	1	$566,045.86	374746056	58.5	21	65300
0.85	1000	2	$564,288.37	498344117	97.5	54	110326
0.85	1000	3	$564,626.91	215873267	110.8	141	126789
0.85	1500	1	$576,335.41	727908059	84.1	36	95250
0.85	1500	2	$566,811.86	178765455	135.5	100	155175
0.85	1500	3	$564,773.70	202118740	185.2	94	212933
0.85	2000	1	$563,907.10	357868025	111.6	38	127200
0.85	2000	2	$554,866.25	115908753	174.6	470	200785
0.85	2000	3	$556,680.25	183590800	237.8	953	274704
0.85	2500	1	$573,093.67	315534178	135.1	168	155500
0.85	2500	2	$556,603.20	240242531	226.1	345	262816
0.85	2500	3	$554,603.75	188465284	300.4	382	349073
0.90	1000	1	$566,236.77	386696869	86.8	14	98500
0.90	1000	2	$562,688.63	356340184	143.4	64	164448
0.90	1000	3	$558,437.88	121159617	177.4	447	204087
0.90	1500	1	$568,844.97	255232737	126.8	109	145950
0.90	1500	2	$560,708.22	279469904	196.6	307	225908
0.90	1500	3	$553,317.84	106127717	263.9	135	305478
0.90	2000	1	$552,388.18	368681348	163.3	226	188200
0.90	2000	2	$555,283.26	188802721	276.5	851	319653
0.90	2000	3	$555,231.07	199527562	341.7	1605	396142
0.90	2500	1	$560,442.90	178366591	208.1	96	242000
0.90	**2500**	**2**	**$547,332.16**	**125284932**	**324.3**	**1054**	**379211**
0.90	2500	3	$547,878.04	120502377	440.7	773	516715

Table 102. Continued

RATE	ACCEPT	FACTOR	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
0.95	1000	1	$568,193.96	451501614	171.4	320	192900
0.95	1000	2	$546,426.57	204339868	293.5	394	333059
0.95	1000	3	$550,525.17	136335379	381.9	683	434199
0.95	1500	1	$556,627.24	56438115	248.3	499	279750
0.95	1500	2	$556,216.84	213233033	409.0	1160	466287
0.95	1500	3	$553,189.86	347079342	533.1	3140	608967
0.95	2000	1	$552,110.59	280588127	332.0	607	377000
0.95	2000	2	$554,617.69	252131323	652.4	40466	593637
0.95	2000	3	$546,429.14	364579587	675.9	5451	773925
0.95	2500	1	$554,743.72	158715950	413.6	711	471250
0.95	2500	2	$538,967.57	101579443	652.9	3443	745593
0.95	2500	3	$546,870.62	106631994	877.6	15740	988528

Table 103. Genetic Algorithm Parameter Experiment Results (CFLP)

POP	GENS	XOVER	MUTATE	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
20	1000	0.85	0.05	$673,657.93	369757105	94.3	327.9	21000
20	1000	0.95	0.05	$670,615.83	224766285	84.8	0.5	21000
20	1000	1.00	0.05	$678,163.28	297737946	85.5	3.1	21000
20	1500	0.85	0.05	$668,553.15	338047001	159.4	2875.9	31000
20	1500	0.95	0.05	$665,512.60	146763771	127.1	2.8	31000
20	1500	1.00	0.05	$662,378.96	174943259	141.7	278.0	31000
20	2000	0.85	0.05	$663,215.33	359049169	171.3	0.6	41000
20	2000	0.95	0.05	$658,802.55	285566936	171.7	0.8	41000
20	2000	1.00	0.05	$660,967.40	152892984	169.9	1.5	41000
30	1000	0.85	0.05	$674,403.08	383112022	133.7	271.7	31500
30	1000	0.95	0.05	$669,116.47	399097940	129.9	16.5	31500
30	1000	1.00	0.05	$674,653.97	421990300	128.9	4.2	31500
30	1500	0.85	0.05	$665,106.65	383594906	189.6	0.6	46500
30	1500	0.95	0.05	$659,293.14	256810448	193.5	4.0	46500
30	1500	1.00	0.05	$662,738.36	255890824	194.7	55.5	46500
30	2000	0.85	0.05	$656,682.90	392913547	258.1	4.0	61500
30	2000	0.95	0.05	$654,699.61	239779160	260.4	13.1	61500
30	2000	1.00	0.05	$657,380.87	144522550	252.6	2.1	61500
40	1000	0.85	0.05	$676,625.32	767567268	170.4	11.6	42000
40	1000	0.95	0.05	$665,826.25	217441518	186.5	615.0	42000
40	1000	1.00	0.05	$671,268.13	256786747	169.4	0.4	42000
40	1500	0.85	0.05	$659,729.69	471523541	251.0	0.3	62000
40	1500	0.95	0.05	$655,430.66	134267945	258.2	0.2	62000
40	1500	1.00	0.05	$660,420.46	206088834	255.9	0.8	62000
40	**2000**	**0.85**	**0.05**	**$649,614.69**	**528334690**	**334.2**	**0.5**	**82000**
40	2000	0.95	0.05	$653,954.00	74823597	344.1	0.4	82000
40	2000	1.00	0.05	$653,773.29	167413065	336.2	0.9	82000
20	1000	0.85	0.10	$731,348.27	384532535	103.7	387.6	21000
20	1000	0.95	0.10	$735,015.90	573927980	95.9	0.3	21000
20	1000	1.00	0.10	$736,742.62	866214939	96.0	0.5	21000
20	1500	0.85	0.10	$723,928.25	331951161	153.2	512.8	31000
20	1500	0.95	0.10	$723,489.30	170558619	161.3	639.7	31000
20	1500	1.00	0.10	$725,027.18	616853471	170.7	946.9	31000
20	2000	0.85	0.10	$722,166.34	250727322	193.2	0.7	41000
20	2000	0.95	0.10	$718,875.32	195987763	193.4	0.4	41000
20	2000	1.00	0.10	$720,080.70	589743323	191.1	0.7	41000
30	1000	0.85	0.10	$728,994.03	460932352	145.6	1.1	31500
30	1000	0.95	0.10	$727,900.85	299351585	150.2	227.6	31500
30	1000	1.00	0.10	$732,091.41	895079430	145.2	0.9	31500
30	1500	0.85	0.10	$723,164.59	364958647	218.4	36.1	46500
30	1500	0.95	0.10	$720,018.84	163853813	217.6	1.8	46500
30	1500	1.00	0.10	$720,594.58	997008850	221.9	195.1	46500
30	2000	0.85	0.10	$709,328.61	180999945	296.5	4.1	61500
30	2000	0.95	0.10	$716,019.47	163035281	295.5	0.6	61500
30	2000	1.00	0.10	$716,838.89	812753492	287.4	1.4	61500

Table 103. Continued

POP	GENS	XOVER	MUTATE	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
40	1000	0.85	0.10	$735,080.66	680289888	190.5	0.5	42000
40	1000	0.95	0.10	$724,013.24	816675918	190.6	0.8	42000
40	1000	1.00	0.10	$742,376.98	223708936	204.1	570.7	42000
40	1500	0.85	0.10	$720,819.19	222843956	284.6	2.8	62000
40	1500	0.95	0.10	$716,802.98	474199785	292.2	3.3	62000
40	1500	1.00	0.10	$729,904.16	103364248	287.5	0.5	62000
40	2000	0.85	0.10	$712,936.41	348412910	379.4	5.2	82000
40	2000	0.95	0.10	$716,701.17	464220433	388.8	10.5	82000
40	2000	1.00	0.10	$722,662.29	375688383	382.7	0.8	82000

Table 104. Tabu Search Parameter Experiment Results (MP-FLP)

LTM	STM	TLSIZE	Average Cost	Variance of Cost	Average Time (sec)	Variance Of Time	Average Solutions
1	1000	5	$2,689,656.20	953159582	67.6	2	500000
1	1000	7	$2,687,730.10	929182331	67.3	0	500000
1	1000	12	$2,692,391.90	1054872669	67.1	0	500000
1	1500	5	$2,689,290.90	923808173	96.6	0	750000
1	1500	7	$2,684,331.30	773595180	97.1	0	750000
1	1500	12	$2,687,529.60	996207884	96.9	0	750000
1	2000	5	$2,688,961.30	958318831	126.4	0	1000000
1	2000	7	$2,683,899.40	808516075	126.6	0	1000000
1	2000	12	$2,685,261.30	986749137	126.7	0	1000000
1	2500	5	$2,688,305.00	947705220	156.5	0	1250000
1	2500	7	$2,683,317.30	801899344	156.4	0	1250000
1	2500	12	$2,684,274.50	953735934	156.3	0	1250000
1	3000	5	$2,686,318.80	832108425	185.9	0	1500000
1	3000	7	$2,682,962.10	779312334	186.0	0	1500000
1	3000	12	$2,683,326.30	931516125	185.8	0	1500000
5	1000	5	$2,669,238.00	556406868	305.6	1	2500000
5	1000	7	$2,660,110.30	569546331	305.3	0	2500000
5	1000	12	$2,660,723.50	408941617	305.8	0	2500000
5	1500	5	$2,657,142.60	178930543	455.4	3	3750000
5	1500	7	$2,662,039.00	329498372	454.6	2	3750000
5	1500	12	$2,659,664.20	165058189	455.2	3	3750000
5	2000	5	$2,647,724.20	775921966	602.9	0	5000000
5	2000	7	$2,662,198.30	297758084	603.5	0	5000000
5	2000	12	$2,652,665.60	536286024	603.9	1	5000000
5	2500	5	$2,652,991.40	165721081	753.7	7	6250000
5	2500	7	$2,661,085.80	238117306	750.5	2	6250000
5	2500	12	$2,652,607.40	266396349	752.1	2	6250000
5	3000	5	$2,647,957.40	315991818	900.1	2	7500000
5	**3000**	**7**	**$2,640,209.90**	**993884527**	**900.9**	**3**	**7500000**
5	3000	12	$2,642,064.40	298625370	920.8	1756	7500000
10	1000	5	$2,650,281.60	218171719	609.2	2	5000000
10	1000	7	$2,652,297.70	267377770	607.9	1	5000000
10	1000	12	$2,654,375.30	428991386	608.6	3	5000000
10	1500	5	$2,650,135.60	162639722	909.4	6	7500000
10	1500	7	$2,650,034.70	272066359	909.3	6	7500000
10	1500	12	$2,649,805.00	44002201	908.4	4	7500000
10	2000	5	$2,638,553.10	347375504	1198.4	5	10000000
10	2000	7	$2,645,994.20	256382801	1196.6	2	10000000
10	2000	12	$2,637,317.60	320640895	1200.3	4	10000000
10	2500	5	$2,645,926.40	149577554	1493.1	9	12500000
10	2500	7	$2,648,617.00	310371745	1492.8	5	12500000
10	2500	12	$2,646,851.60	70224088	1496.4	0	12500000
10	3000	5	$2,641,990.60	143113622	1794.3	0	15000000
10	3000	7	$2,630,409.60	752689576	1864.9	1	15000000
10	3000	12	$2,637,154.90	267433168	1795.3	3	15000000

Table 105. Simulated Annealing Parameter Experiment Results (MP-FLP)

RATE	ACCEPT	FACTOR	Average Cost	Variance of Cost	Average Time (sec)	Variance Of Time	Average Solutions
0.75	1000	1	$2,743,301.60	2248365246	50.3	8	37400
0.75	1000	2	$2,651,235.40	871325881	70.6	6	60744
0.75	1000	3	$2,634,034.20	381396150	84.7	15	77238
0.75	1500	1	$2,684,845.20	1112734978	64.7	6	54450
0.75	1500	2	$2,640,291.50	667996283	92.1	11	86740
0.75	1500	3	$2,596,559.00	595634814	114.5	13	112942
0.75	2000	1	$2,654,448.10	821394161	78.1	2	70000
0.75	2000	2	$2,618,753.60	440842860	115.3	6	114094
0.75	2000	3	$2,595,898.40	581161381	143.0	14	147708
0.75	2500	1	$2,635,911.30	496126318	92.8	11	87500
0.75	2500	2	$2,605,652.50	685280760	139.4	14	143669
0.75	2500	3	$2,582,664.70	321122474	170.7	16	181799
0.8	1000	1	$2,717,325.80	953949876	57.7	2	46100
0.8	1000	2	$2,643,000.50	608321762	83.0	4	75160
0.8	1000	3	$2,607,132.50	731974532	100.1	8	95566
0.8	1500	1	$2,664,895.70	887525278	75.6	10	67050
0.8	1500	2	$2,607,326.10	1004728402	111.6	6	109858
0.8	1500	3	$2,590,485.70	455826310	139.2	14	143177
0.8	2000	1	$2,647,808.90	656535079	92.5	6	87400
0.8	2000	2	$2,600,885.60	496022504	139.3	25	143206
0.8	2000	3	$2,576,414.30	191951698	175.8	50	187249
0.8	2500	1	$2,634,470.50	638747165	112.1	11	110500
0.8	2500	2	$2,582,131.10	741621909	167.8	12	178538
0.8	2500	3	$2,558,100.00	608948570	210.1	33	229788
0.85	1000	1	$2,678,514.90	420723637	91.7	111	60000
0.85	1000	2	$2,617,621.30	1009070118	101.7	12	97647
0.85	1000	3	$2,608,119.00	637101035	125.8	7	127436
0.85	1500	1	$2,654,300.40	980519532	92.7	6	87450
0.85	1500	2	$2,597,636.90	346500713	141.2	14	146564
0.85	1500	3	$2,588,538.50	525984280	176.0	32	189120
0.85	2000	1	$2,625,367.70	313507607	117.9	7	118000
0.85	2000	2	$2,584,013.40	216833832	178.4	14	192097
0.85	2000	3	$2,564,482.60	835518578	224.1	57	248597
0.85	2500	1	$2,620,338.70	449266793	139.9	18	145250
0.85	2500	2	$2,566,307.00	680225624	227.1	687	236390
0.85	2500	3	$2,548,291.70	143335891	270.6	72	305747
0.9	1000	1	$2,639,148.70	522420903	93.5	8	87200
0.9	1000	2	$2,612,918.20	602314090	143.6	7	146765
0.9	1000	3	$2,566,201.70	267011503	179.1	16	189257
0.9	1500	1	$2,612,715.80	427025922	131.4	21	132300
0.9	1500	2	$2,579,739.20	159656418	204.4	19	219728
0.9	1500	3	$2,560,128.10	484893829	258.2	46	284300
0.9	2000	1	$2,600,432.10	490174967	165.3	10	173200
0.9	2000	2	$2,567,205.30	501091931	268.1	78	296660
0.9	2000	3	$2,538,557.20	345305299	331.6	65	373248
0.9	2500	1	$2,587,648.80	368643527	201.1	25	215750
0.9	2500	2	$2,550,907.20	203736991	321.6	72	361315
0.9	**2500**	**3**	**$2,535,707.10**	**165251077**	**407.5**	**104**	**465027**

Table 105. Continued

RATE	ACCEPT	FACTOR	Average Cost	Variance of Cost	Average Time (sec)	Variance Of Time	Average Solutions
0.95	1000	1	$2,591,426.10	468311605	162.7	25	170300
0.95	1000	2	$2,556,468.20	279613538	261.7	45	289452
0.95	1000	3	$2,545,407.50	376702523	328.6	113	370259
0.95	1500	1	$2,576,283.50	183454622	235.6	23	258150
0.95	1500	2	$2,557,959.80	734671387	380.6	23	432941
0.95	1500	3	$2,540,453.40	209124976	489.2	69	564749
0.95	2000	1	$2,560,206.10	365580138	310.8	62	348600
0.95	2000	2	$2,536,430.80	179890935	498.1	31	575078
0.95	2000	3	$2,525,710.00	397681137	626.7	319	731124
0.95	2500	1	$2,551,920.80	213500059	376.2	48	428000
0.95	2500	2	$2,524,155.60	134162224	611.5	69	712471
0.95	2500	3	$2,499,668.90	174981449	775.8	155	911122

Table 106. Genetic Algorithm Parameter Experiment Results (MP-FLP)

POP	GENS	XOVER	MUTATE	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
20	1000	0.80	0.05	$3,232,619	578117377	208	2.3	18020
20	1000	0.85	0.05	$3,245,770	3027169458	209	2.0	18020
20	1000	0.95	0.05	$3,228,012	2064199771	207	2.2	18020
20	1500	0.80	0.05	$2,967,841	1012390933	317	2.1	27020
20	1500	0.85	0.05	$2,972,513	1467582983	309	6.0	27020
20	1500	0.95	0.05	$2,973,628	1119396645	311	4.9	27020
20	2000	0.80	0.05	$2,859,617	1141305642	409	10.2	36020
20	2000	0.85	0.05	$2,861,498	598305973	409	5.2	36020
20	2000	0.95	0.05	$2,869,377	928857653	410	9.7	36020
30	1000	0.80	0.05	$3,233,953	2598211786	308	0.5	28030
30	1000	0.85	0.05	$3,220,733	1330011718	308	2.4	28030
30	1000	0.95	0.05	$3,211,496	1822860761	309	0.4	28030
30	1500	0.80	0.05	$2,970,380	778327549	463	1.2	42030
30	1500	0.85	0.05	$2,975,501	894947800	461	1.1	42030
30	1500	0.95	0.05	$2,979,679	981189044	466	6.1	42030
30	**2000**	**0.80**	**0.05**	**$2,856,778**	**805514932**	**614**	**11.2**	**56030**
30	2000	0.85	0.05	$2,873,624	1054606524	613	13.0	56030
30	2000	0.95	0.05	$2,879,959	166769566	614	15.7	56030
40	1000	0.80	0.05	$3,215,423	982939817	412	1.9	38040
40	1000	0.85	0.05	$3,194,845	1922156731	414	3.1	38040
40	1000	0.95	0.05	$3,171,852	2242594477	416	50.3	38040
40	1500	0.80	0.05	$2,982,460	832197559	612	1.2	57040
40	1500	0.85	0.05	$2,983,578	505746231	612	2.1	57040
40	1500	0.95	0.05	$2,940,615	1049269754	616	0.4	57040
40	2000	0.80	0.05	$2,888,219	727430883	815	1.0	76040
40	2000	0.85	0.05	$2,877,486	1119599640	822	1.2	76040
40	2000	0.95	0.05	$2,850,135	1532001055	823	3.1	76040
20	1000	0.80	0.10	$3,276,601	2603965932	209	1.8	18020
20	1000	0.85	0.10	$3,242,772	2740490348	211	2.4	18020
20	1000	0.95	0.10	$3,226,137	1750499418	209	1.7	18020
20	1500	0.80	0.10	$3,071,168	1546219449	319	2.1	27020
20	1500	0.85	0.10	$3,049,118	2332904556	313	11.9	27020
20	1500	0.95	0.10	$3,039,553	1196184611	313	3.6	27020
20	2000	0.80	0.10	$2,965,226	1562503212	413	14.3	36020
20	2000	0.85	0.10	$2,952,160	1001035487	412	2.5	36020
20	2000	0.95	0.10	$2,950,261	1666382596	412	8.7	36020
30	1000	0.80	0.10	$3,275,489	1843485238	311	0.4	28030
30	1000	0.85	0.10	$3,241,267	1126397875	311	0.7	28030
30	1000	0.95	0.10	$3,240,682	696093810	312	0.4	28030
30	1500	0.80	0.10	$3,083,701	2812656267	467	1.7	42030
30	1500	0.85	0.10	$3,053,420	778719640	466	1.1	42030
30	1500	0.95	0.10	$3,071,568	717422860	470	7.5	42030
30	2000	0.80	0.10	$2,980,739	1947004452	621	12.4	56030
30	2000	0.85	0.10	$2,971,632	527987509	619	9.7	56030
30	2000	0.95	0.10	$2,985,438	1235808570	621	17.6	56030

Table 106. Continued

POP	GENS	XOVER	MUTATE	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
40	1000	0.80	0.10	$3,228,536	2149915082	416	4.4	38040
40	1000	0.85	0.10	$3,245,007	1680126786	417	3.6	38040
40	1000	0.95	0.10	$3,242,197	931791686	417	17.3	38040
40	1500	0.80	0.10	$3,052,240	1240341367	617	0.1	57040
40	1500	0.85	0.10	$3,073,512	942090114	617	0.6	57040
40	1500	0.95	0.10	$3,073,290	1279806716	622	0.8	57040
40	2000	0.80	0.10	$2,976,248	806766101	825	0.4	76040
40	2000	0.85	0.10	$2,986,423	835252960	832	1.0	76040
40	2000	0.95	0.10	$2,988,599	1259665420	834	13.1	76040

Table 107. Tabu Search Parameter Experiment Results (MC-FLP)

LTM	STM	TLSIZE	Average Cost	Variance of Cost	Average Time (sec)	Variance Of Time	Average Solutions
1	25	1	$2,718,055.60	394993076788	88.9	43	481
1	25	2	$2,715,572.20	393352131516	88.6	32	480
1	25	5	$2,716,897.40	394693381798	88.3	37	480
1	50	1	$2,715,432.20	394108663965	191.7	253	981
1	50	2	$2,712,790.60	390052008103	193.3	209	980
1	50	5	$2,711,178.70	388423936656	190.8	178	980
1	75	1	$2,714,574.50	395272854197	292.7	655	1481
1	75	2	$2,711,996.60	389003584548	296.7	490	1480
1	75	5	$2,711,150.10	388462658730	293.1	457	1480
1	100	1	$2,713,387.50	393327466708	395.4	1212	1980
1	100	2	$2,711,182.30	390102295924	399.6	846	1980
1	100	5	$2,705,674.60	385171494282	394.8	833	1980
3	25	1	$1,957,825.80	13467084275	272.7	154	1449
3	25	2	$2,206,649.30	232808380507	271.2	141	1448
3	25	5	$2,180,993.00	234182581947	264.5	59	1448
3	50	1	$2,069,268.90	133868246306	576.0	704	2946
3	50	2	$2,079,745.70	139156919336	583.9	591	2949
3	50	5	$2,201,197.40	141918528286	571.3	399	2945
3	75	1	$2,298,626.00	288599102004	889.4	1312	4452
3	75	2	$2,042,154.60	121144298046	879.4	745	4450
3	75	5	$2,271,966.00	259294494745	890.1	784	4449
3	100	1	$2,049,598.70	149115226974	1170.6	1949	5950
3	100	2	$2,160,553.20	222835245687	1215.2	2283	5949
3	100	5	$2,126,992.90	268713734339	1193.4	1544	5950
5	25	1	$1,886,119.50	11182800371	450.7	240	2420
5	25	2	$2,080,064.90	137554718838	451.8	219	2417
5	25	5	$1,900,044.90	10554348630	448.0	159	2416
5	50	1	$1,962,688.30	47004949590	962.2	771	4911
5	**50**	**2**	**$1,877,646.30**	**12613949678**	**968.6**	**470**	**4916**
5	50	5	$2,005,743.00	113899703010	963.8	550	4914
5	75	1	$2,007,560.30	75157849438	1490.2	990	7413
5	75	2	$1,888,053.30	6064224750	1480.6	634	7421
5	75	5	$1,944,790.40	116889633183	1492.2	942	7420
5	100	1	$1,877,393.70	7656274956	2014.5	9169	9922
5	100	2	$2,052,390.90	126834009181	2019.6	4198	9917
5	100	5	$1,845,173.30	8301716283	2016.6	3004	9922

Table 108. Simulated Annealing Parameter Experiment Results (MC-FLP)

RATE	ACCEPT	FACTOR	Average Cost	Variance of Cost	Average Time (sec)	Variance Of Time	Average Solutions
0.75	25	1	$2,714,217.60	397870021480	82.7	52	783
0.75	25	2	$2,713,679.70	393295489123	151.1	67	1413
0.75	25	3	$2,709,277.00	386898981293	195.6	337	1765
0.75	50	1	$2,699,935.10	399886655458	175.0	396	1635
0.75	50	2	$2,703,539.60	397144480569	302.6	1679	2773
0.75	50	3	$2,706,974.30	395270255426	367.9	1163	3333
0.75	75	1	$2,700,989.90	390234705053	267.9	2198	2490
0.75	75	2	$2,700,791.90	394978872731	446.9	1963	4093
0.75	75	3	$2,696,195.50	395602379007	580.1	4669	5181
0.8	25	1	$2,708,313.40	400640220526	104.4	99	988
0.8	25	2	$2,714,691.30	398248261275	183.9	112	1712
0.8	25	3	$2,709,167.50	390518894343	231.3	1407	2101
0.8	50	1	$2,714,258.20	404436632189	207.9	126	1950
0.8	50	2	$2,702,145.00	396003911013	373.3	793	3422
0.8	50	3	$2,703,657.40	391850196738	490.7	4550	4439
0.8	75	1	$2,705,628.10	391449346630	314.9	1905	2970
0.8	75	2	$2,696,050.60	395197781134	587.4	2442	5388
0.8	75	3	$2,696,346.50	393764217007	726.3	8123	6530
0.85	25	1	$2,712,095.40	404432176216	131.5	279	1253
0.85	25	2	$2,704,726.90	395217649694	252.2	562	2326
0.85	25	3	$2,702,873.50	398892659872	319.0	1276	2913
0.85	50	1	$2,702,008.20	402020310204	286.2	308	2665
0.85	50	2	$2,703,315.00	406015948104	494.7	966	4539
0.85	50	3	$2,696,661.80	400947976774	642.9	2673	5819
0.85	75	1	$2,702,831.00	393712699120	435.3	1419	4028
0.85	75	2	$2,699,766.00	397613748198	768.8	10102	7078
0.85	75	3	$2,697,139.90	400262228090	963.4	11810	8654
0.9	25	1	$2,704,685.70	392708986614	200.6	200	1910
0.9	25	2	$2,712,438.00	388199649494	365.4	775	3399
0.9	25	3	$2,702,678.70	402003039640	456.4	1595	4143
0.9	50	1	$2,702,781.90	393576218774	417.4	866	3920
0.9	50	2	$2,702,544.70	397867707671	760.1	1536	6929
0.9	**50**	**3**	**$2,696,056.60**	**399359073882**	**932.4**	**2921**	**8448**
0.9	75	1	$2,704,922.00	393498099523	650.8	2424	6098
0.9	75	2	$2,689,265.10	397352652912	1177.2	7670	10850
0.9	75	3	$2,688,925.70	400115745384	1438.6	18987	12977
0.95	25	1	$2,705,616.90	400083698529	378.6	727	3640
0.95	25	2	$2,694,720.50	391029689967	722.9	701	6768
0.95	25	3	$2,694,154.10	392975877636	914.6	14943	8373
0.95	50	1	$2,700,987.30	386604853237	831.9	1044	7840
0.95	50	2	$2,691,272.90	395051163064	1461.0	7934	13573
0.95	50	3	$2,693,115.30	390849371765	1850.5	25106	16836
0.95	75	1	$2,696,688.50	389246338251	1293.9	5275	12083
0.95	75	2	$2,690,430.40	398562977292	2257.8	9522	20931
0.95	75	3	$2,688,180.00	395832348443	2926.7	52011	26600

Table 109. Genetic Algorithm Parameter Experiment Results (MC-FLP)

POP	GENS	XOVER	MUTATE	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
20	50	0.80	0.05	$2,174,957.40	23948035886	104	38.1	920
20	50	0.85	0.05	$2,196,908.80	8311521603	103	38.0	920
20	50	0.95	0.05	$2,315,908.50	102444343361	103	22.1	920
20	100	0.80	0.05	$1,981,202.60	11625621810	249	214.0	1820
20	100	0.85	0.05	$1,937,109.00	4243067099	250	221.5	1820
20	100	0.95	0.05	$1,954,617.50	8371165081	244	115.0	1820
20	150	0.80	0.05	$1,873,157.40	4127853222	408	333.0	2720
20	150	0.85	0.05	$1,893,309.10	4387466710	408	701.0	2720
20	150	0.95	0.05	$1,824,466.40	4269614035	406	228.8	2720
30	50	0.80	0.05	$2,125,382.70	10459530317	161	121.5	1430
30	50	0.85	0.05	$2,108,950.10	7715275793	157	45.9	1430
30	50	0.95	0.05	$2,099,763.70	8960922987	156	42.8	1430
30	100	0.80	0.05	$1,903,301.90	5818789231	375	1180.7	2830
30	100	0.85	0.05	$1,885,550.70	5961597176	370	327.1	2830
30	100	0.95	0.05	$1,877,819.70	6875785186	371	391.8	2830
30	150	0.80	0.05	$1,845,335.90	4835958053	609	2907.7	4230
30	150	0.85	0.05	$1,840,309.30	5543624656	609	1335.4	4230
30	150	0.95	0.05	$1,820,773.40	5196735514	607	694.4	4230
40	50	0.80	0.05	$2,030,628.90	14035346965	208	97.4	1940
40	50	0.85	0.05	$2,075,180.50	16372468700	202	82.1	1940
40	50	0.95	0.05	$2,026,939.50	10669993187	211	116.4	1940
40	100	0.80	0.05	$1,899,681.10	4989726288	487	847.7	3840
40	100	0.85	0.05	$1,870,159.40	8441749222	478	652.2	3840
40	100	0.95	0.05	$1,853,945.10	3197391043	494	1167.6	3840
40	150	0.80	0.05	$1,838,477.90	4082903267	802	2244.5	5740
40	150	0.85	0.05	$1,801,685.60	6146428738	785	1411.6	5740
40	150	0.95	0.05	$1,802,761.60	2354821426	805	3866.4	5740
20	50	0.80	0.10	$1,979,709.20	12040756201	111	39.2	920
20	50	0.85	0.10	$2,006,990.50	9985173069	111	55.8	920
20	50	0.95	0.10	$1,956,368.20	6593559836	116	89.5	920
20	100	0.80	0.10	$1,834,156.30	5059227981	259	219.0	1820
20	100	0.85	0.10	$1,837,010.90	4674762152	264	129.1	1820
20	100	0.95	0.10	$1,828,914.60	4231305612	275	415.9	1820
20	150	0.80	0.10	$1,794,110.20	3516937542	418	532.9	2720
20	150	0.85	0.10	$1,796,442.70	2917865229	422	216.9	2720
20	150	0.95	0.10	$1,801,375.20	2994060159	439	601.1	2720
30	50	0.80	0.10	$1,963,066.10	7169053812	170	223.3	1430
30	50	0.85	0.10	$2,005,921.30	10906716460	167	103.1	1430
30	50	0.95	0.10	$2,159,415.40	116169883847	171	64.4	1430
30	100	0.80	0.10	$1,826,936.40	4253714398	390	720.8	2830
30	100	0.85	0.10	$1,851,729.30	3005811954	388	485.3	2830
30	100	0.95	0.10	$1,873,957.00	3282560296	395	322.2	2830
30	150	0.80	0.10	$1,795,509.10	1874218209	627	1143.3	4230
30	150	0.85	0.10	$1,823,633.40	1801401543	625	1210.3	4230
30	150	0.95	0.10	$1,813,255.00	2796754313	634	862.0	4230

Table 109. Continued

POP	GENS	XOVER	MUTATE	Average Cost	Variance of Cost	Average Time	Variance Of Time	Average Solutions
40	50	0.80	0.10	$1,951,525.40	9266085136	225	156.9	1940
40	50	0.85	0.10	$1,991,455.60	5556420765	217	130.3	1940
40	50	0.95	0.10	$1,890,418.40	5239644294	225	378.4	1940
40	100	0.80	0.10	$1,827,258.30	3130696592	535	1207.6	3840
40	100	0.85	0.10	$1,847,441.50	2336746112	512	773.9	3840
40	100	0.95	0.10	$1,809,680.30	2249412933	523	766.1	3840
40	150	0.80	0.10	$1,788,680.80	2320875949	859	2320.7	5740
40	150	0.85	0.10	$1,801,082.90	1622034169	829	1310.8	5740
40	**150**	**0.95**	**0.10**	**$1,779,198.90**	**1630572981**	**830**	**1269.5**	**5740**

APPENDIX C

U.S. CITIES DATABASE FOR TEST PROBLEMS

The table on the next page contains a list of 250 cities in the United States that were selected to form a database for generating a new set of test problems for this research. The city locations, latitude and longitude were obtained from *The World Almanac and Book of Facts.* The estimated 1996 population for these cities was obtained from the *Rand McNally Commercial Atlas & Marketing Guide 1997.*

In order to build our database, it was necessary to obtain the distance between any two of these cities. Since there was no readily available source with this information, we used the latitude and longitude information to compute the line-of-sight distance. This distance was computed using a piece of software freely available from the U.S. Geological Survey. Given the latitude and longitude of two points on the earth, the software computes the distance in statue miles taking into account the curvature of the globe. Although this is not as accurate as the true driving distance, it is a good estimate of the relative distances between the cities.

Table 110. 250 U.S. Cities Database

City	1996 POP	Latitude	Longitude
Abilene,TX	110100	032:027:005N	099:043:051W
Akron,OH	221300	041:005:000N	081:030:044W
Albany,NY	107000	042:039:001N	073:045:001W
Albuquerque,NM	419300	035:005:001N	106:039:005W
Allentown,PA	130900	040:036:011N	075:028:006W
Amarillo,TX	174400	035:012:027N	101:050:004W
Anchorage,AK	257600	061:010:000N	149:059:000W
AnnArbor,MI	109300	042:016:059N	083:044:052W
Ashland,KY	24400	038:028:036N	082:038:023W
Atlanta,GA	402000	033:045:010N	084:023:037W
AtlanticCity,NJ	35400	039:021:032N	074:025:053W
Augusta,GA	42300	033:028:020N	081:058:000W
Austin,TX	529000	030:016:009N	097:044:037W
Bakersfield,CA	196500	035:022:031N	119:001:018W
Baltimore,MD	686900	039:017:026N	076:036:045W
Bangor,ME	32200	044:048:013N	068:046:018W
BatonRouge,LA	228300	030:026:058N	091:011:000W
BattleCreek,MI	55300	042:018:058N	085:010:048W
Beaumont,TX	114800	030:005:020N	094:006:009W
Berkeley,CA	101200	037:052:109N	122:016:017W
Bethlehem,PA	73200	040:037:016N	075:022:034W
Billings,MT	87800	045:047:000N	108:030:004W
Biloxi,MS	48900	030:023:048N	088:053:000W
Birmingham,AL	263800	033:031:001N	086:048:036W
Bismarck,ND	53900	046:048:023N	100:047:017W
Bloomington,IL	56500	040:028:058N	088:059:036W
Boise,ID	153400	043:037:007N	116:011:058W
Boston,MA	546000	042:021:024N	071:003:025W
BowlingGreen,KY	46600	036:059:041N	086:026:033W
Bridgeport,CT	130500	041:010:049N	073:011:022W
Brownsville,TX	121500	025:054:007N	097:029:058W
Buffalo,NY	304200	042:052:052N	078:052:021W
Burlington,VT	38500	044:028:034N	073:012:046W
Butte,MN	34400	046:001:006N	112:032:011W
Cambridge,MA	101500	042:022:001N	071:006:022W
Camden,NJ	80000	039:056:041N	075:007:014W
Canton,OH	84100	040:047:050N	081:022:037W
CarsonCity,NV	47100	039:010:000N	119:046:000W
CedarRapids,IA	115000	041:058:001N	091:039:053W
Champaign,IL	68300	040:007:005N	088:014:048W
Charleston,SC	80900	032:046:035N	079:055:053W
Charleston,WV	56000	038:021:001N	081:037:052W
Charlotte,NC	456700	035:013:044N	080:050:045W
Chattanooga,TN	151600	035:002:041N	085:018:032W
Cheyenne,WY	54400	041:008:009N	104:049:007W
Chicago,IL	2708000	041:052:028N	087:038:022W

Cincinnati,OH	352800	039:006:007N	084:030:035W
Cleveland,OH	485600	041:029:051N	081:041:050W
ColoradoSprings,CO	330300	038:050:007N	104:049:016W
Columbia,MO	75700	038:057:003N	092:019:046W
Columbia,SC	107200	034:000:002N	081:002:000W
Columbus,GA	186800	032:028:007N	084:059:024W
Columbus,OH	633200	039:057:047N	083:000:017W
Concord,NH	36600	043:012:022N	071:032:025W
CorpusChristi,TX	281300	027:047:051N	097:023:045W
Dallas,TX	1033600	032:047:009N	096:047:037W
Davenport,IA	96900	041:031:019N	090:034:033W
Dayton,OH	176300	039:045:032N	084:011:043W
DaytonaBeach,FL	64300	029:012:044N	081:001:010W
Decatur,IL	81900	039:050:042N	088:056:047W
Denver,CO	499700	039:044:058N	104:059:022W
DesMoines,IA	194300	041:035:014N	093:037:000W
Detroit,MI	979900	042:019:048N	083:002:057W
DodgeCity,KS	22800	037:045:017N	100:001:009W
Duluth,MN	83100	046:046:056N	092:006:024W
Durham,NC	145800	036:000:000N	078:054:045W
ElPaso,TX	601700	031:045:036N	106:029:011W
Enid,OK	46300	036:023:040N	097:052:035W
Erie,PA	108400	042:007:015N	080:004:057W
Eugene,OR	118700	044:003:016N	123:005:030W
Eureka,CA	28000	040:048:008N	124:009:046W
Evansville,IN	130200	037:058:020N	087:034:021W
Fairbanks,AK	35000	064:048:000N	147:051:000W
FallRiver,MA	88500	041:042:006N	071:009:018W
Fargo,ND	81600	046:052:030N	096:047:018W
Flagstaff,AZ	52900	035:011:036N	111:039:006W
Flint,MI	137000	043:000:050N	083:041:033W
FortWayne,IN	186600	041:004:021N	085:008:026W
FortWorth,TX	462000	032:044:055N	097:019:044W
Fresno,CA	408000	036:044:012N	119:047:011W
FtSmith,AR	74700	035:023:010N	094:025:036W
Gainesville,FL	89500	029:038:056N	082:019:019W
Gallup,NM	20100	035:031:030N	108:044:030W
Galveston,TX	58700	029:018:010N	094:047:043W
Gary,IN	112300	041:036:012N	087:020:019W
GrandJunction,CO	37000	039:004:006N	108:033:054W
GrandRapids,MI	190100	042:058:003N	085:040:013W
GreatFalls,MN	59100	047:029:033N	111:018:023W
GreenBay,WI	103800	044:030:048N	088:000:050W
Greensboro,NC	199800	036:004:017N	079:047:025W
Greenville,SC	60100	034:050:050N	082:024:001W
Gulfport,MS	78600	030:022:004N	089:005:036W
Harrisburg,PA	55000	040:015:043N	076:052:059W
Hartford,CT	118400	041:046:012N	072:040:049W
Helena,MN	26500	046:035:033N	112:002:024W

Hilo,HI	43400	019:043:030N	155:005:024W
Honolulu,HI	385600	021:018:022N	157:051:035W
Houston,TX	1710600	029:045:026N	095:021:037W
Huntington,WV	53200	038:025:012N	082:026:033W
Huntsville,AL	159900	034:044:018N	086:035:019W
Indianapolis,IN	759200	039:046:007N	086:009:046W
IowaCity,IA	61300	041:039:037N	091:031:053W
Jackson,MI	38700	042:014:043N	084:024:022W
Jackson,MS	191900	032:017:056N	090:011:006W
Jacksonville,FL	681400	030:019:044N	081:039:042W
JerseyCity,NJ	225900	040:043:050N	074:003:056W
Johnstown,PA	27200	040:019:035N	078:055:003W
Joplin,MO	43900	037:005:026N	094:030:000W
Juneau,AK	28600	058:018:012N	134:024:030W
Kalamazoo,MI	81400	042:017:029N	085:035:014W
KansasCity,KS	447200	039:007:004N	094:038:024W
KansasCity,MO	85800	039:004:056N	094:035:020W
Kenosha,WI	24000	042:035:043N	087:050:011W
KeyWest,FL	174500	024:033:030N	081:048:012W
Knoxville,TN	47300	035:057:039N	083:055:007W
Lafayette,IN	58300	040:025:011N	086:053:039W
Lancaster,PA	116000	040:002:025N	076:018:029W
Lansing,MI	159400	042:044:001N	084:033:015W
Laredo,TX	366400	027:030:022N	099:030:030W
LasVegas,NV	242400	036:010:020N	115:008:037W
Lexington,KY	43900	038:002:050N	084:029:046W
Lima,OH	207700	040:044:035N	084:006:020W
Lincoln,NE	177800	040:048:059N	096:042:015W
LittleRock,AR	432500	034:044:042N	092:016:037W
LongBeach,CA	3420500	033:046:014N	118:011:018W
LosAngeles,CA	268100	034:003:015N	118:014:028W
Louisville,KY	94100	038:014:047N	085:045:049W
Lowell,MA	197600	042:038:025N	071:019:014W
Lubbock,TX	109500	033:035:005N	101:050:033W
Macon,GA	195600	032:050:012N	083:037:036W
Madison,WI	96600	043:004:023N	089:022:055W
Manchester,NH	23300	042:059:028N	071:027:041W
Marshall,TX	613400	032:033:000N	094:023:000W
Memphis,TN	376000	035:008:046N	090:003:013W
Miami,FL	613300	025:046:037N	080:011:032W
Milwaukee,WI	350800	043:002:019N	087:054:015W
Mineapolis,MN	36100	044:058:057N	093:015:043W
Minot,ND	206900	048:014:009N	101:017:038W
Mobile,AL	71200	030:041:036N	888:002:033W
Muncie,IN	513100	040:011:028N	085:023:016W
Nashville,TN	257200	036:009:033N	086:046:055W
Newark,NJ	93200	040:044:014N	074:010:019W
NewBedford,MA	116300	041:038:013N	070:055:041W
NewHaven,CT	481000	041:018:025N	072:055:030W

NewOrleans,LA	7313800	029:056:053N	090:004:010W
NewYork,NY	59400	040:045:006N	073:059:039W
NiagaraFalls,NY	3500	043:005:034N	079:003:026W
Nome,AK	237300	064:030:000N	165:025:000W
Norfolk,VA	373100	036:051:010N	076:017:021W
Oakland,CA	68600	037:048:003N	122:015:054W
Ogden,UT	467600	041:013:031N	111:058:021W
OklahomaCity,OK	349700	035:028:026N	097:031:004W
Omaha,NE	183200	041:015:042N	095:056:014W
Orlando,FL	26800	028:032:042N	081:022:038W
Paducah,KY	136500	037:005:013N	088:035:056W
Pasadena,CA	136400	034:008:044N	118:008:041W
Paterson,NJ	61300	040:055:001N	074:010:021W
Pensacola,FL	111500	030:024:051N	087:012:056W
Peoria,IL	1503000	040:041:042N	089:035:033W
Philadelphia,PA	1088200	039:056:058N	075:009:021W
Phoenix,AZ	351500	033:027:012N	112:004:028W
Pittsburgh,PA	58300	040:026:019N	080:000:000W
PortArthur,TX	61700	029:052:030N	093:056:015W
Portland,ME	457400	043:039:033N	070:015:019W
Portland,OR	19100	045:031:006N	122:040:035W
Portsmouth,NH	102600	043:004:030N	070:045:024W
Portsmouth,VA	147700	036:050:007N	076:018:014W
Providence,RI	86900	041:049:032N	071:024:041W
Provo,UT	101300	040:014:006N	111:039:024W
Pueblo,CO	85800	048:016:017N	104:036:033W
Racine,WI	247200	042:043:049N	087:047:012W
Raleigh,NC	58000	035:046:038N	078:038:021W
RapidCity,SD	77500	044:004:052N	103:013:011W
Reading,PA	150100	040:020:009N	075:055:040W
Reno,NV	200700	039:031:027N	119:048:040W
Richmond,VA	96600	037:032:015N	077:026:009W
Roanoke,VA	77100	037:016:013N	079:056:044W
Rochester,MN	228500	044:001:021N	092:028:003W
Rochester,NY	144900	043:009:041N	077:036:021W
Rockford,IL	383600	042:016:007N	089:005:048W
Sacramento,CA	147900	038:034:057N	121:029:041W
Saginaw,MI	118000	043:025:052N	083:056:005W
Salem,OR	44400	044:056:024N	123:001:059W
Salina,KN	175000	038:050:036N	097:036:046W
SaltLakeCity,UT	91000	040:045:023N	111:053:026W
SanAngelo,TX	1025300	031:027:039N	100:026:003W
SanAntonio,TX	198200	029:025:037N	098:029:006W
SanBernardino,CA	1181900	034:006:030N	117:017:028W
SanDiego,CA	749100	032:042:053N	117:009:021W
SanFrancisco,CA	841300	037:046:039N	122:024:040W
SanJose,CA	443372	037:020:016N	121:053:024W
SanJuan,PR	89300	018:027:000N	066:004:015W
SantaBarbara,CA	48600	034:025:018N	119:041:055W

SantaCruz,CA	64300	036:058:018N	122:001:018W
SantaFe,NM	140700	035:041:011N	105:056:010W
Savannah,GA	63200	032:004:042N	081:005:037W
Schenectady,NY	76500	042:048:042N	073:055:042W
Scranton,PA	522500	041:024:032N	075:039:046W
Seattle,WA	50700	047:036:032N	122:020:012W
Sheboygan,WI	15300	043:045:003N	087:042:052W
Sheridan,WY	196600	044:047:055N	106:057:010W
Shreveport,LA	83300	032:030:046N	093:044:058W
SiouxCity,IA	111500	042:029:046N	096:024:030W
SiouxFalls,SD	67500	043:032:035N	096:043:035W
Somerville,MA	104400	042:023:015N	071:006:007W
SouthBend,IN	45700	041:040:033N	086:015:001W
Spartanburg,SC	196400	034:057:003N	081:056:006W
Spokane,WA	105400	047:039:032N	117:025:033W
Springfield,IL	154600	039:047:058N	089:038:051W
Springfield,MO	70100	037:013:003N	093:017:032W
Springfield,OH	107500	039:055:038N	083:048:029W
Stamford,CT	355600	041:003:009N	073:032:024W
StLouis,MO	229500	038:037:045N	090:012:022W
Stockton,CA	257000	037:057:030N	121:017:016W
StPaul,MN	239500	044:057:019N	093:006:007W
StPetersburgh,FL	27400	027:046:018N	082:038:019W
Superior,WI	156700	046:043:014N	092:006:007W
Syracuse,NY	182900	043:003:004N	076:009:014W
Tacoma,WA	137000	047:014:059N	122:026:015W
Tallahassee,FL	286800	030:026:030N	084:016:056W
Tampa,FL	61600	027:056:058N	082:027:025W
TerreHaute,IN	33300	039:028:003N	087:024:026W
Texarkana,TX	319100	033:025:048N	094:002:030W
Toledo,OH	120200	041:039:014N	083:032:039W
Topeka,KS	82400	039:003:016N	095:040:023W
Trenton,NJ	5100	040:013:014N	074:046:013W
Troy,NY	451500	042:043:045N	073:040:058W
Tucson,AZ	372600	032:013:015N	110:058:008W
Tulsa,OK	38000	036:009:012N	095:059:034W
Urbana,IL	62300	040:006:042N	088:012:006W
Utica,NY	107200	043:006:012N	075:013:033W
Waco,TX	29400	031:033:012N	097:008:000W
WallaWalla,WA	547900	046:004:008N	118:020:024W
Washington,DC	100900	038:053:051N	077:000:033W
Waterbury,CT	65800	041:033:013N	073:002:031W
Waterloo,IA	78100	041:029:040N	092:020:020W
WestPalmBeach,FL	33500	026:042:036N	080:003:007W
Wheeling,WV	49600	040:004:003N	080:043:020W
WhitePlains,NY	307500	041:002:000N	073:045:048W
Wichita,KS	97700	037:041:030N	097:020:016W
WichitaFalls,TX	72600	033:054:034N	098:029:028W
Wilmington,DE	64900	039:044:046N	075:032:051W

Wilmington,NC	156000	034:014:014N	077:056:058W
WinstonSalem,NC	165700	036:005:052N	080:014:042W
Worcester,MA	67500	042:015:037N	071:048:017W
Yakima,WA	180700	046:036:009N	120:030:039W
Yonkers,NY	46800	040:055:055N	073:053:054W
York,PA	89500	039:057:035N	076:043:036W
Youngstown,OH	71300	041:005:057N	080:039:002W
Yuma,AZ	27700	032:042:054N	114:037:024W

APPENDIX D

BOXPLOT CHARTS OF PARAMETER EFFECTS

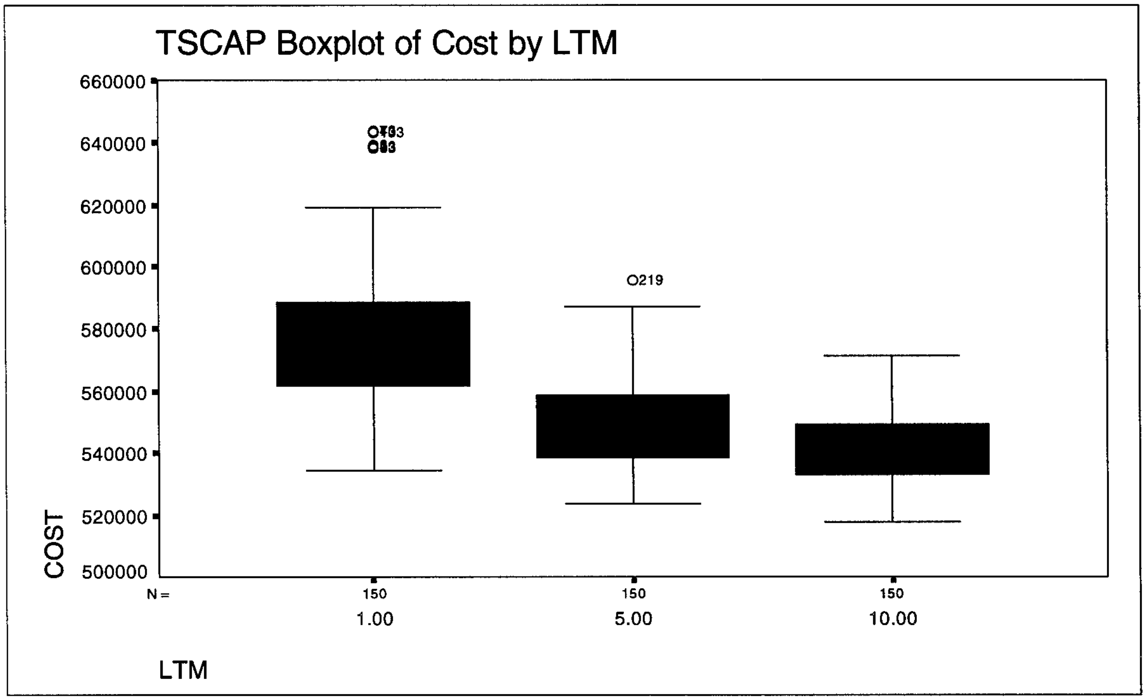

Figure 60. TSCAP Boxplot of Cost by LTM

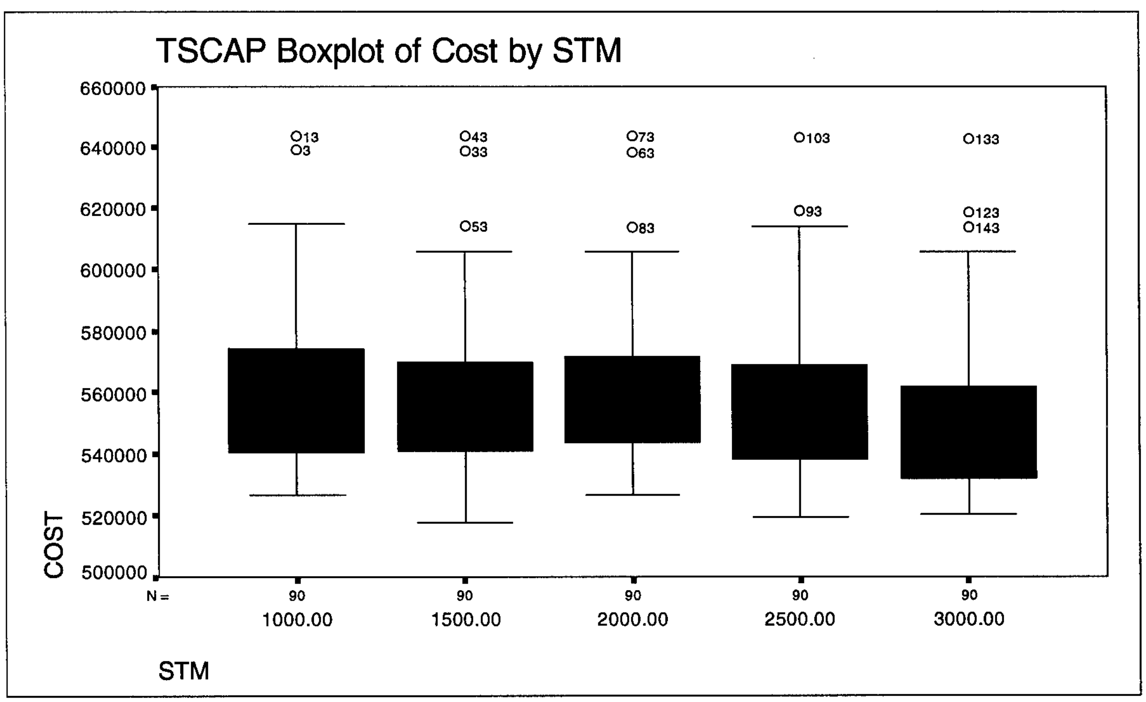

Figure 61. TSCAP Boxplot of Cost by STM

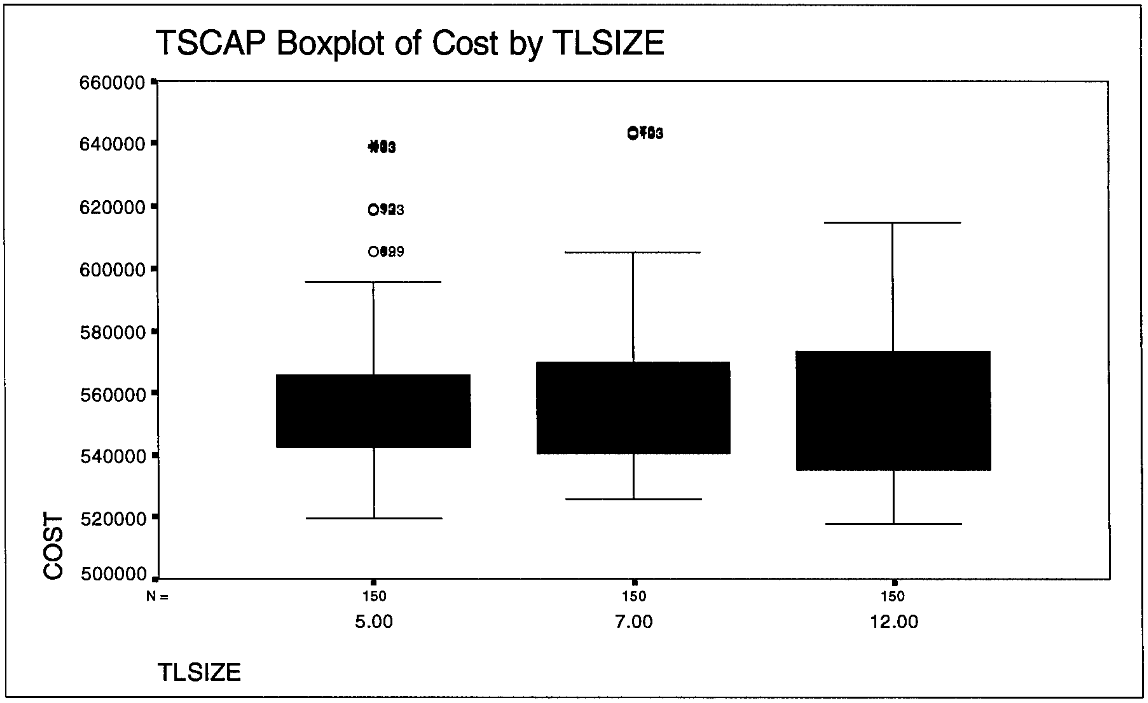

Figure 62. TSCAP Boxplot of Cost by TLSIZE

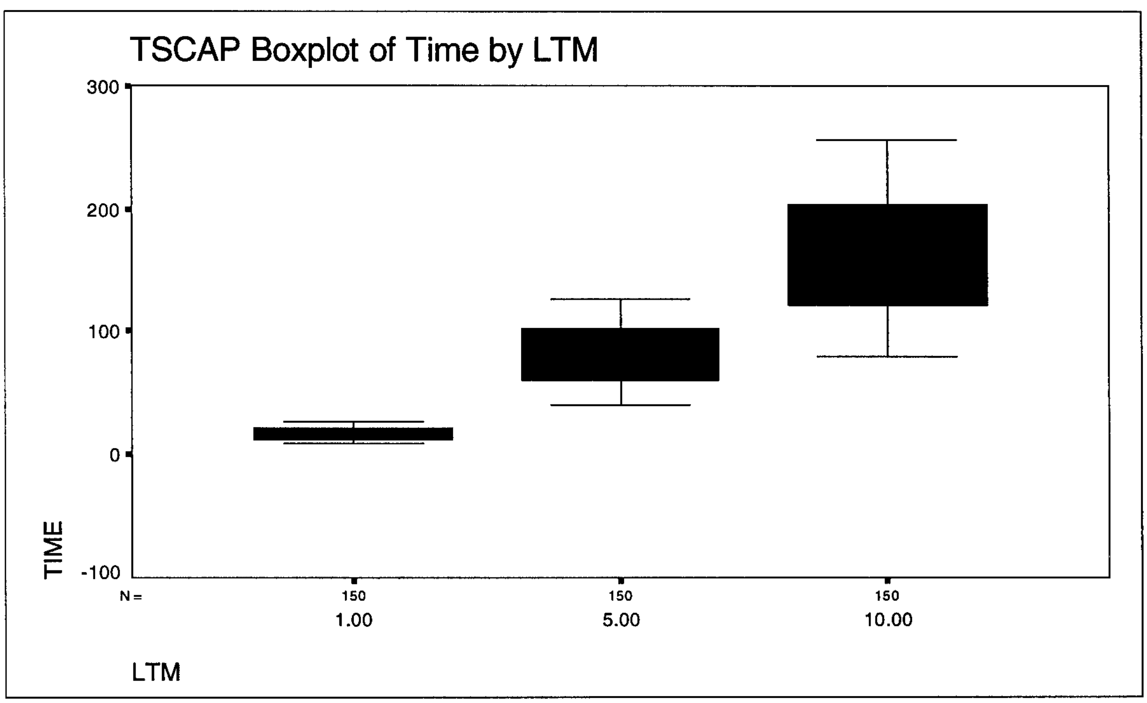

Figure 63. TSCAP Boxplot of Time by LTM

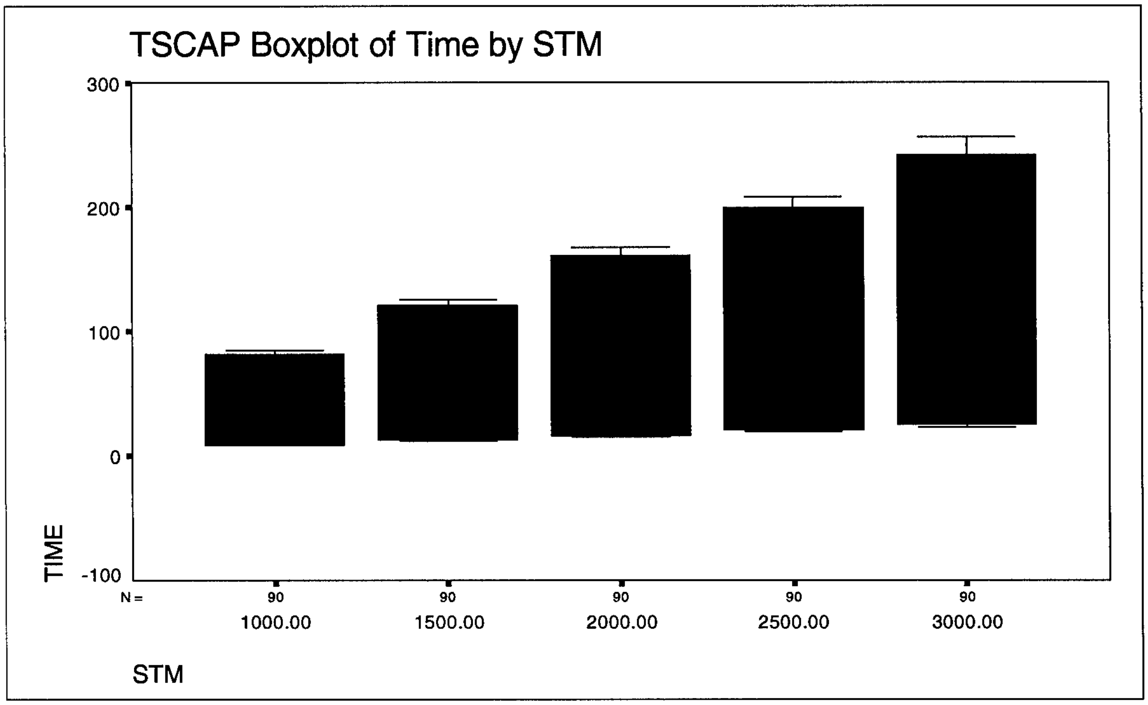

Figure 64. TSCAP Boxplot of Time by STM

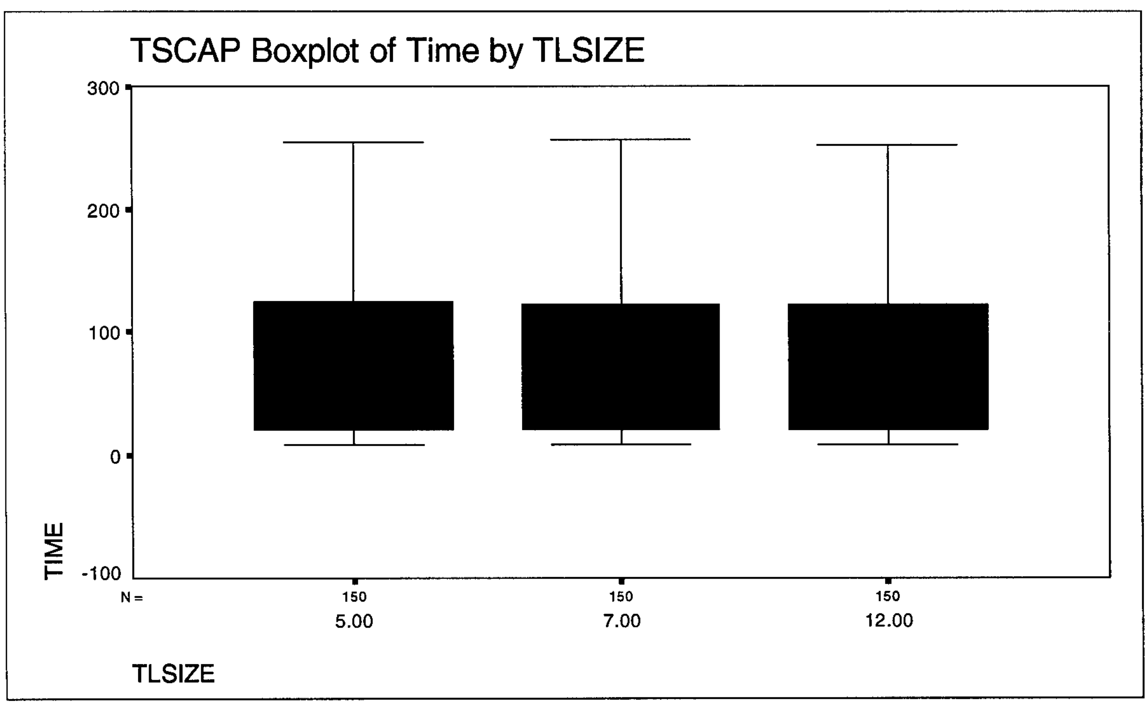

Figure 65. TSCAP Boxplot of Time by TLSIZE

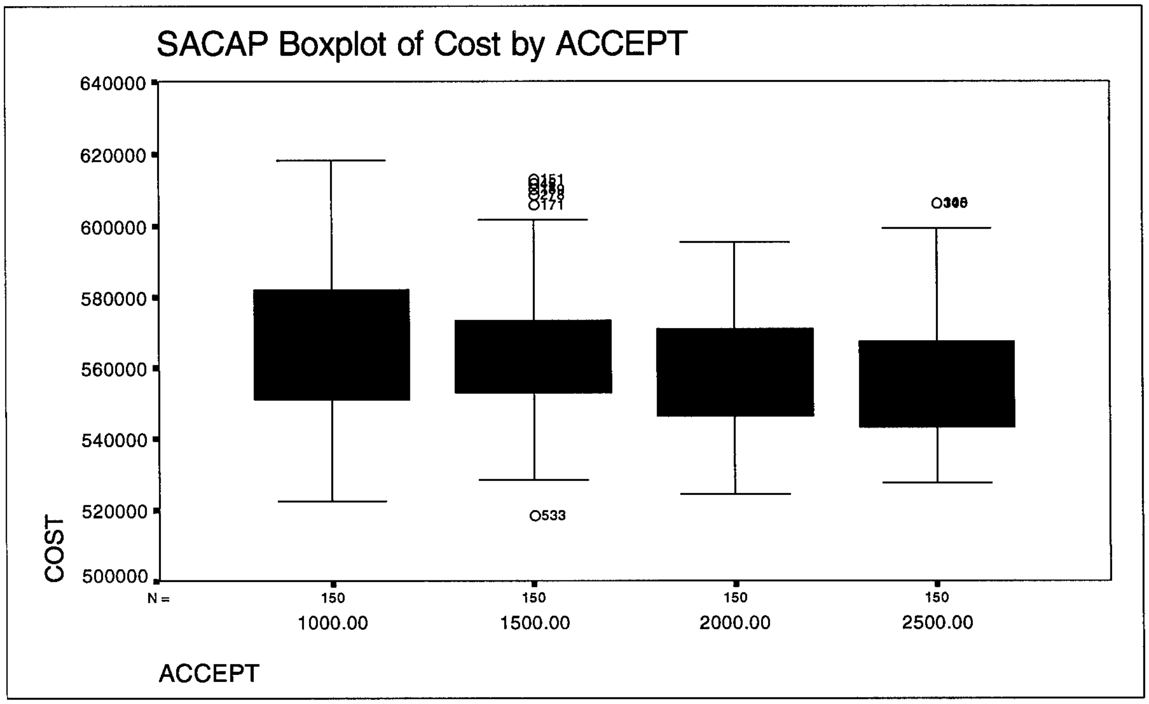

Figure 66. SACAP Boxplot of Cost by ACCEPT

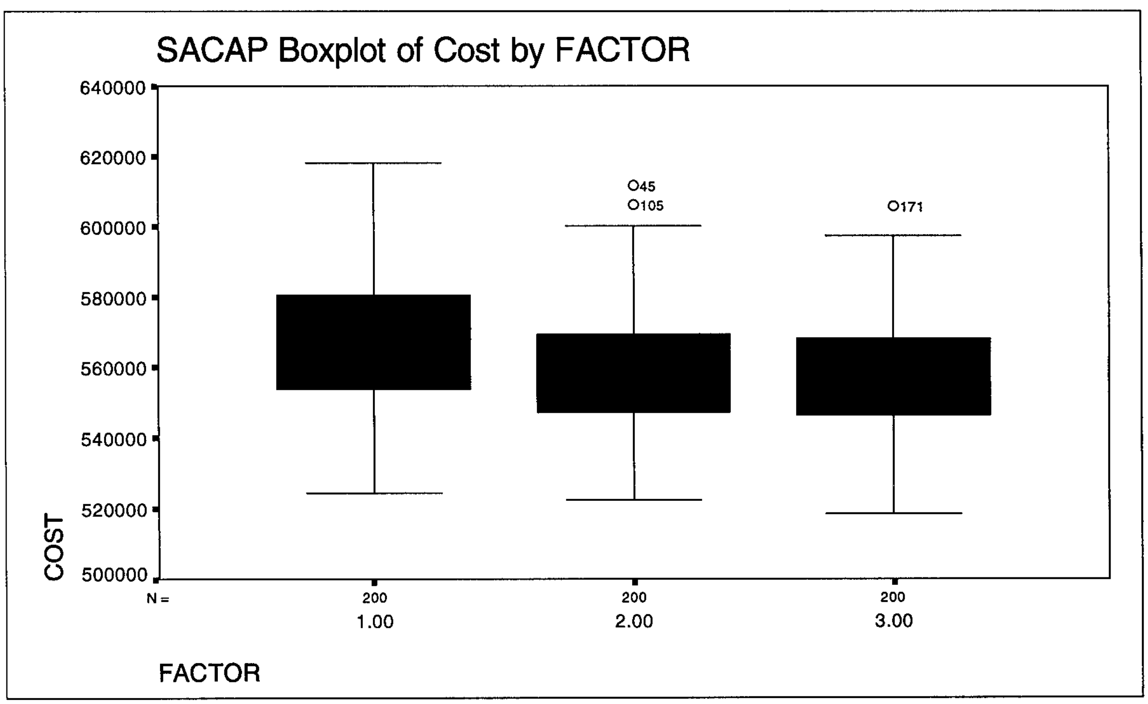

Figure 67. SACAP Boxplot of Cost by FACTOR

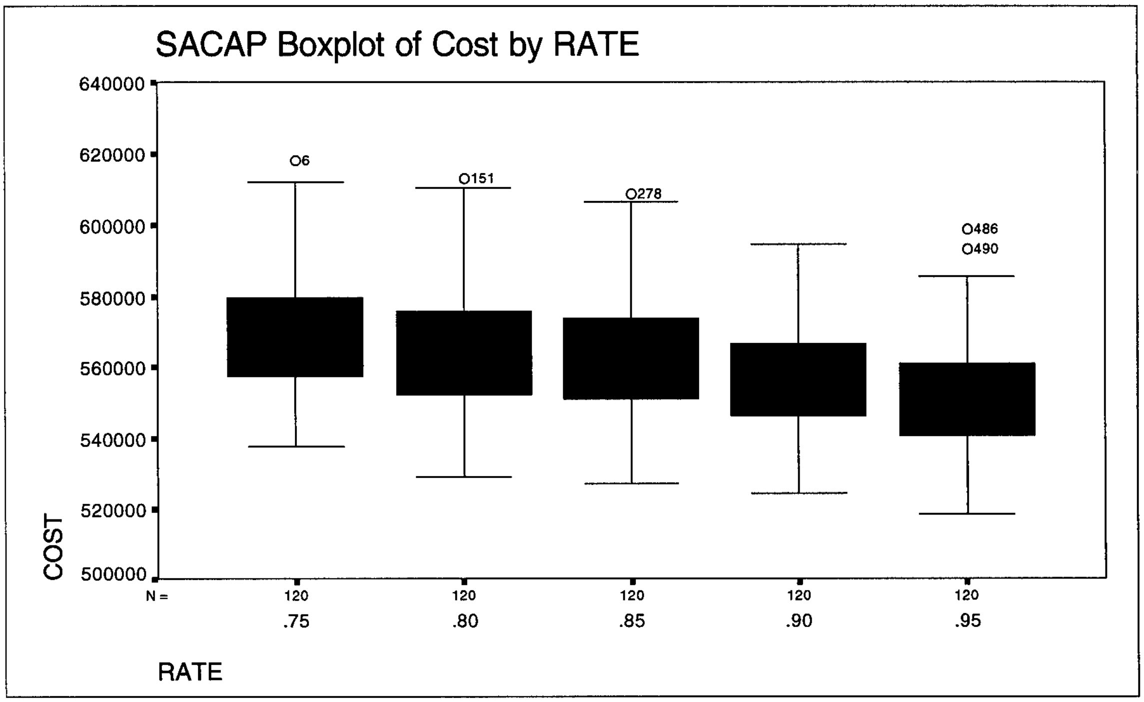

Figure 68. SACAP Boxplot of Cost by RATE

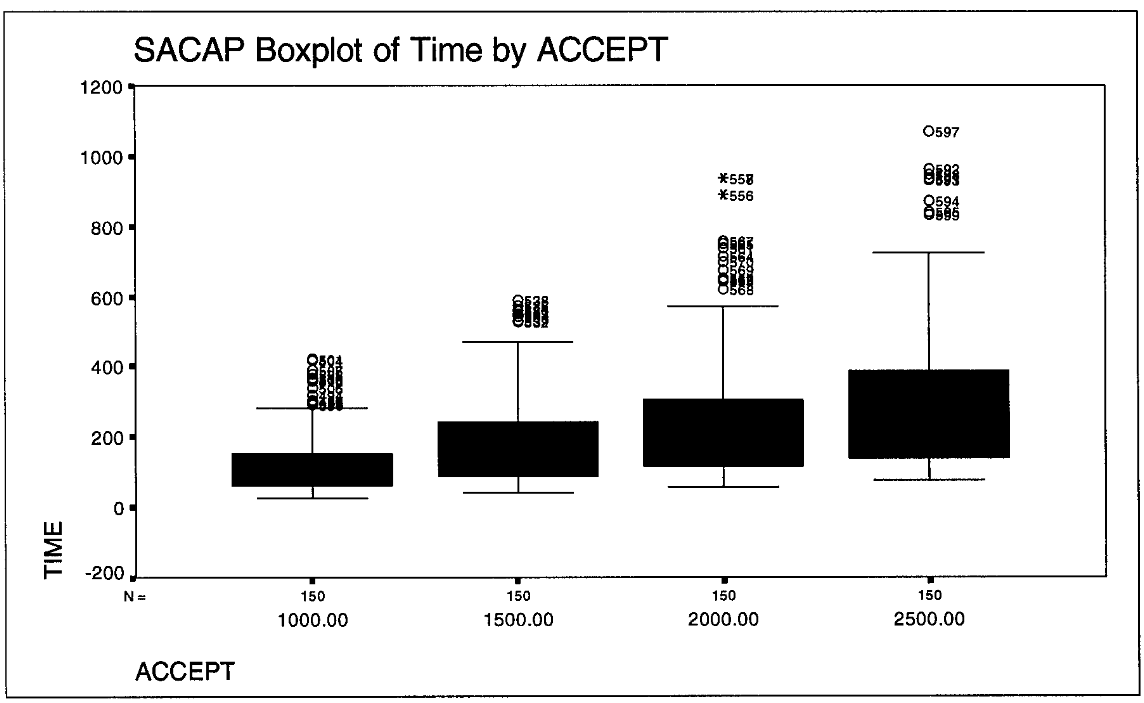

Figure 69. SACAP Boxplot of Time by ACCEPT

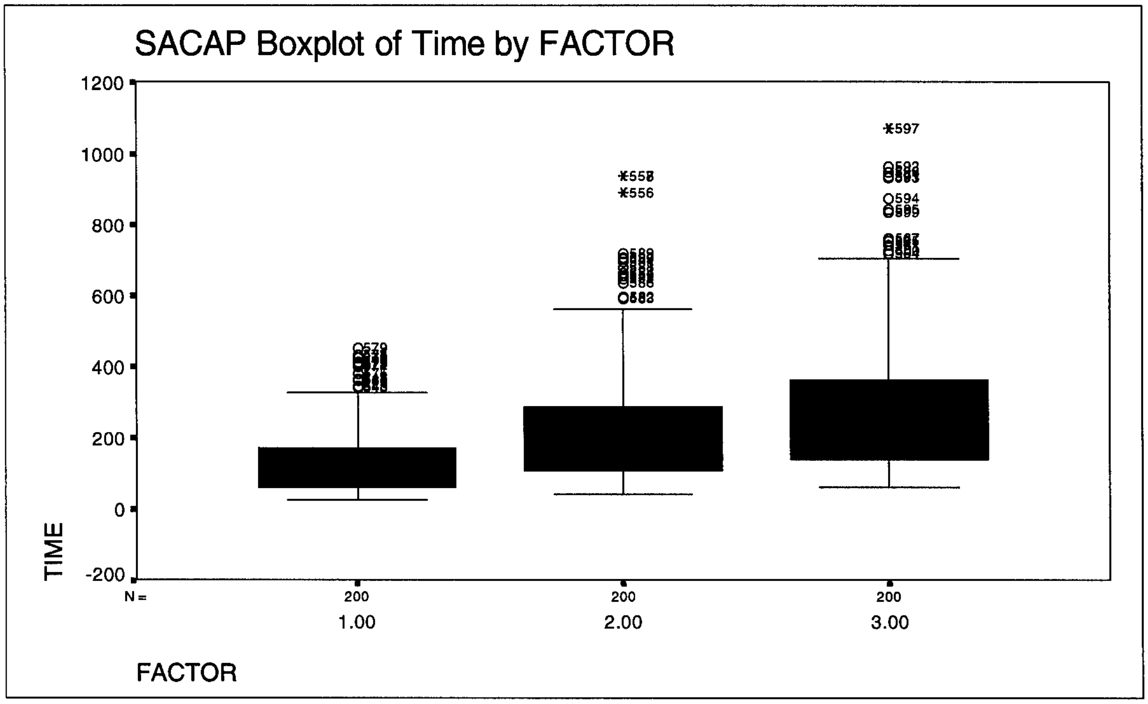

Figure 70. SACAP Boxplot of Time by FACTOR

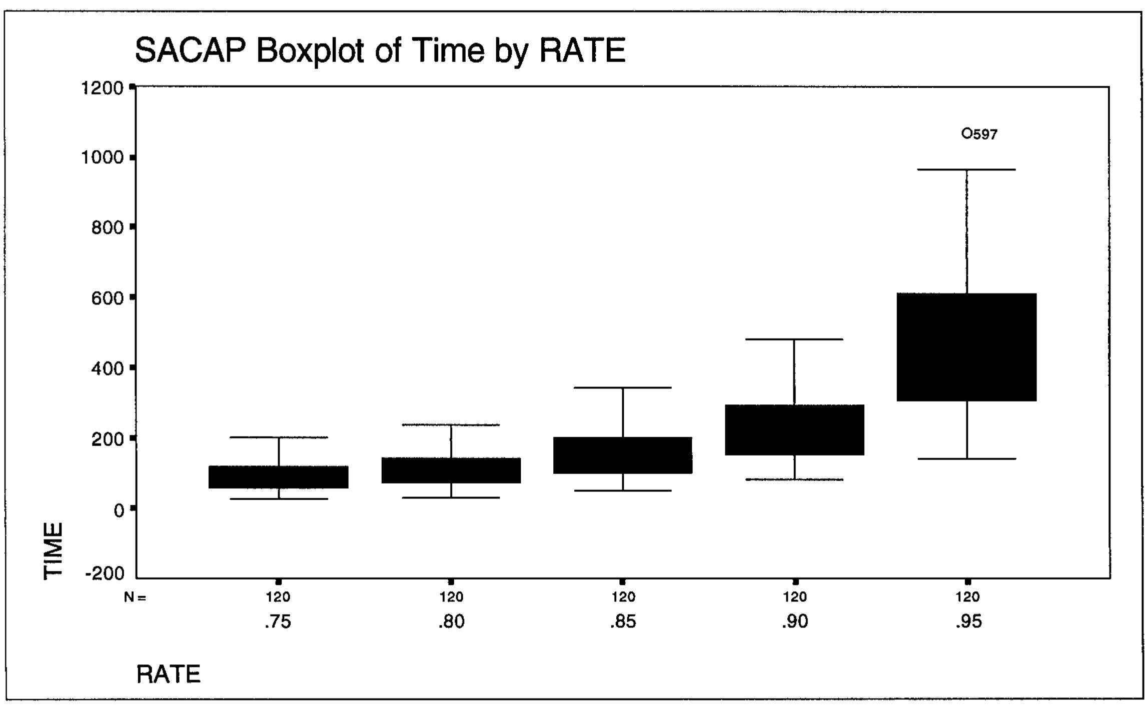

Figure 71. SACAP Boxplot of Time by RATE

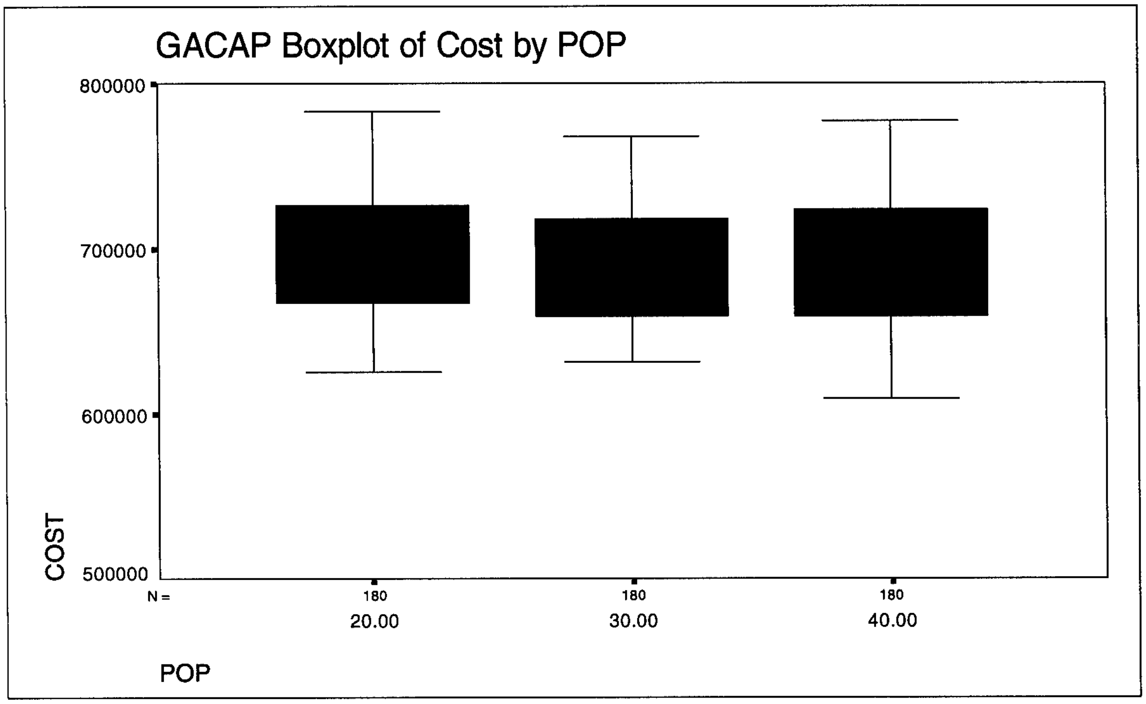

Figure 72. GACAP Boxplot of Cost by POP

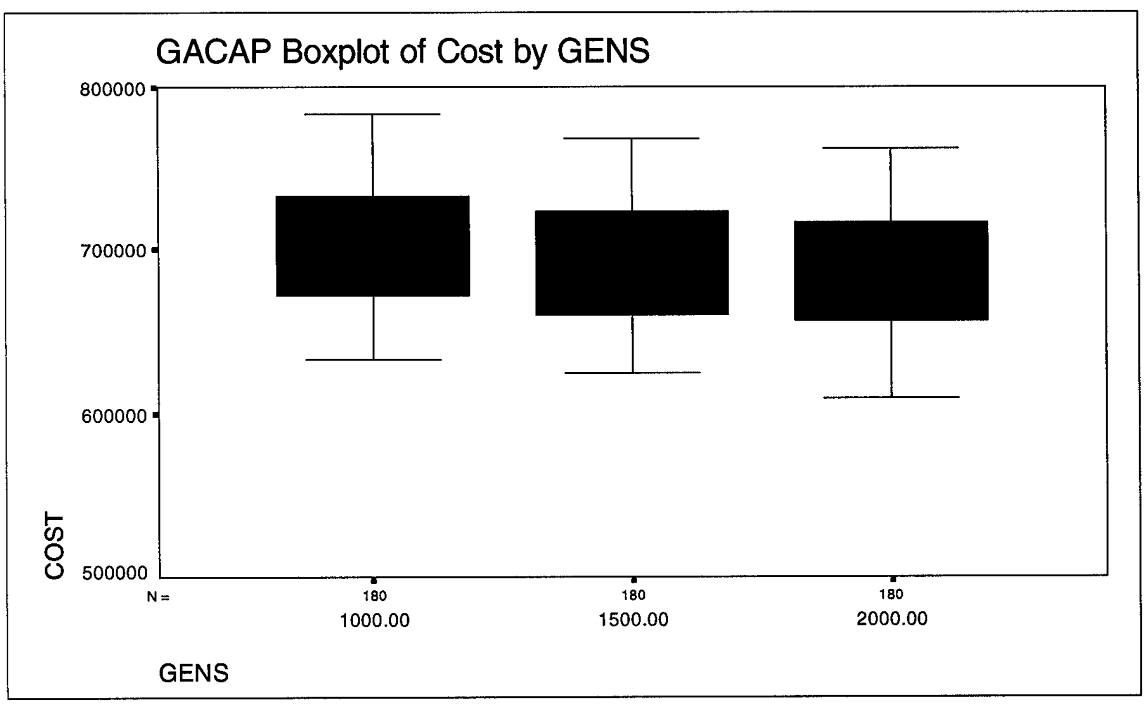

Figure 73. GACAP Boxplot of Cost by GENS

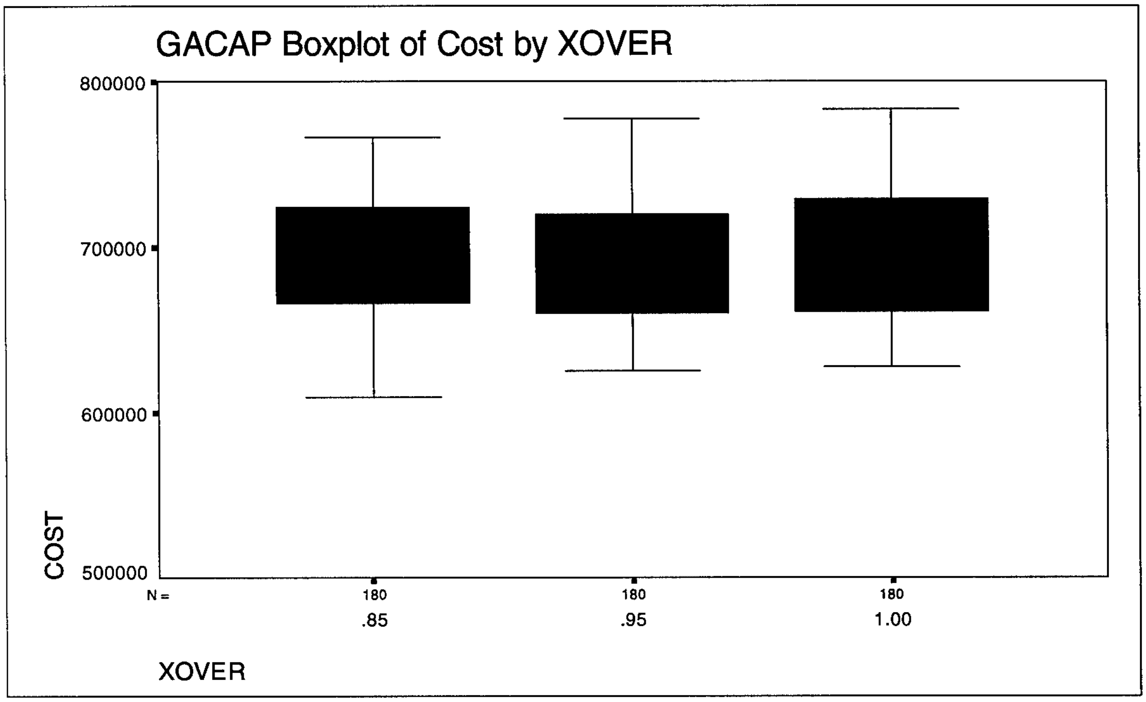

Figure 74. GACAP Boxplot of Cost by XOVER

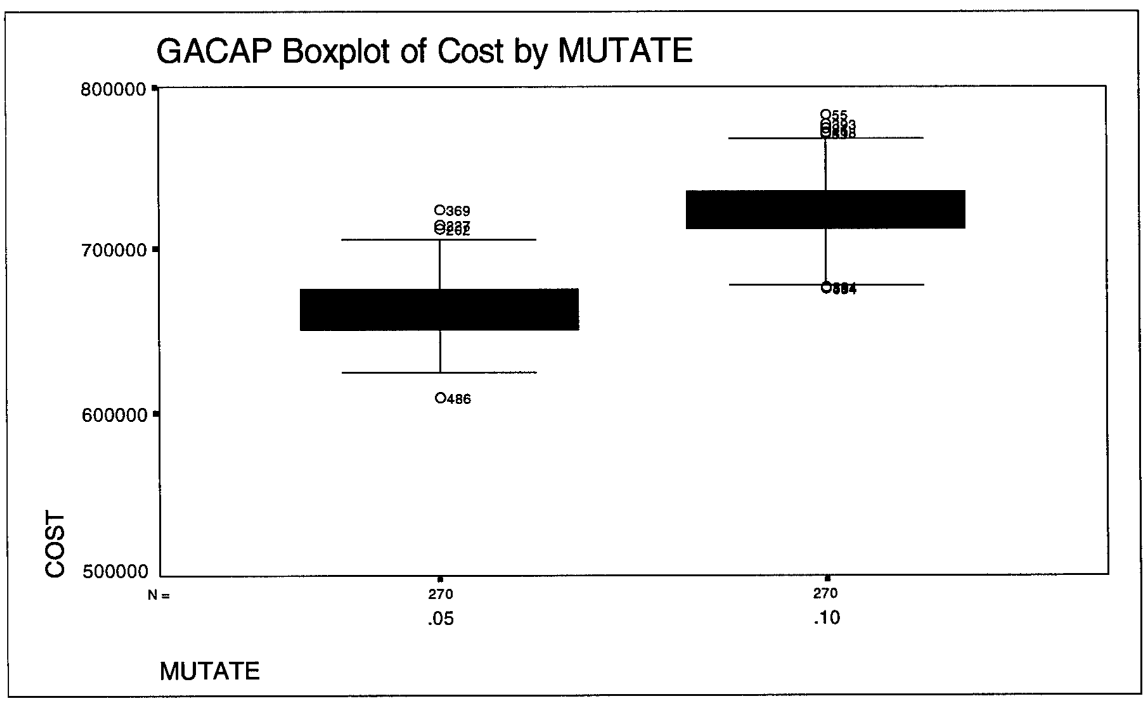

Figure 75. GACAP Boxplot of Cost by MUTATE

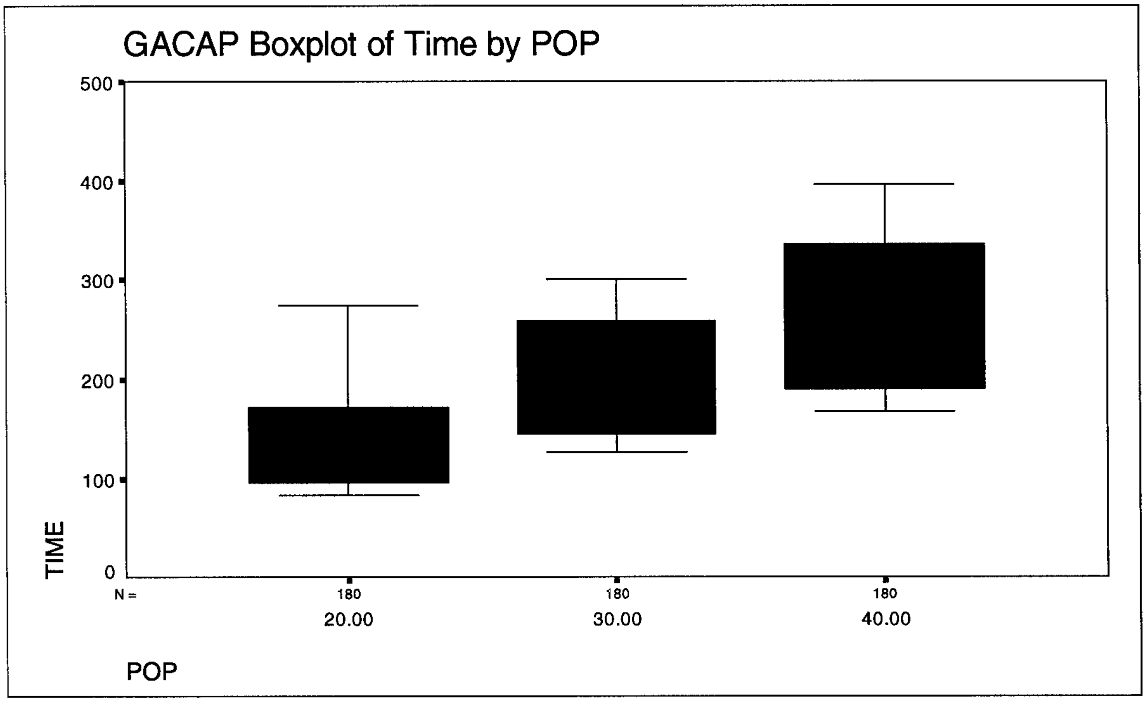

Figure 76. GACAP Boxplot of Time by POP

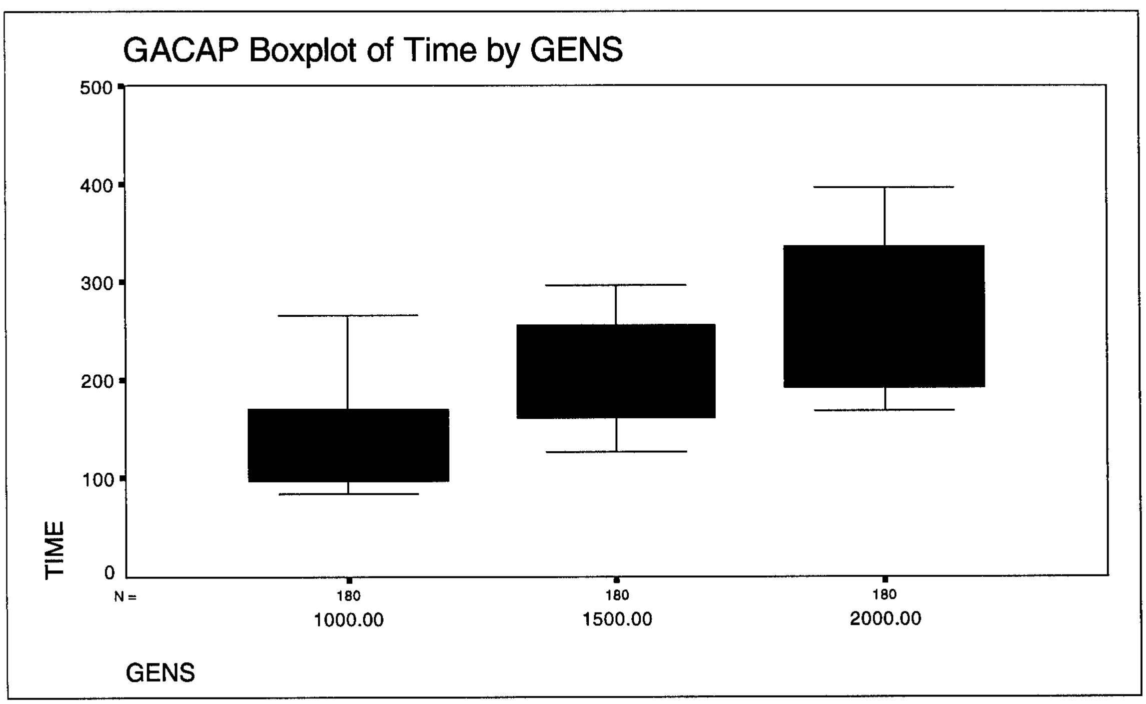

Figure 77. GACAP Boxplot of Time by GENS

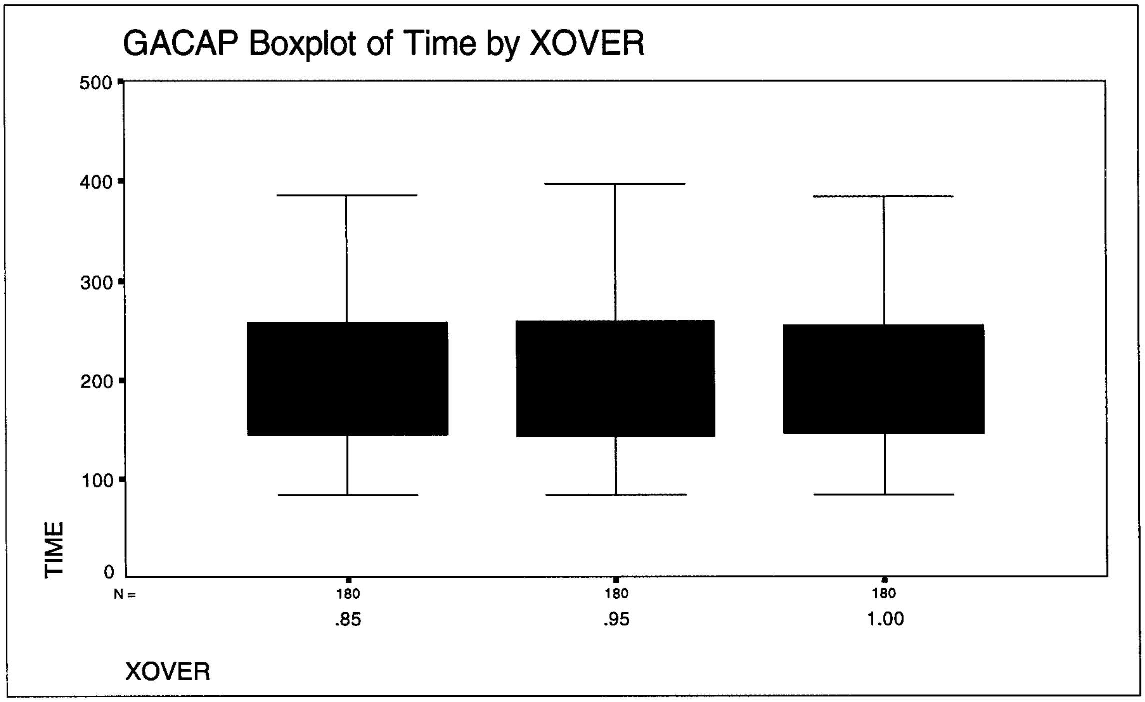

Figure 78. GACAP Boxplot of Time by XOVER

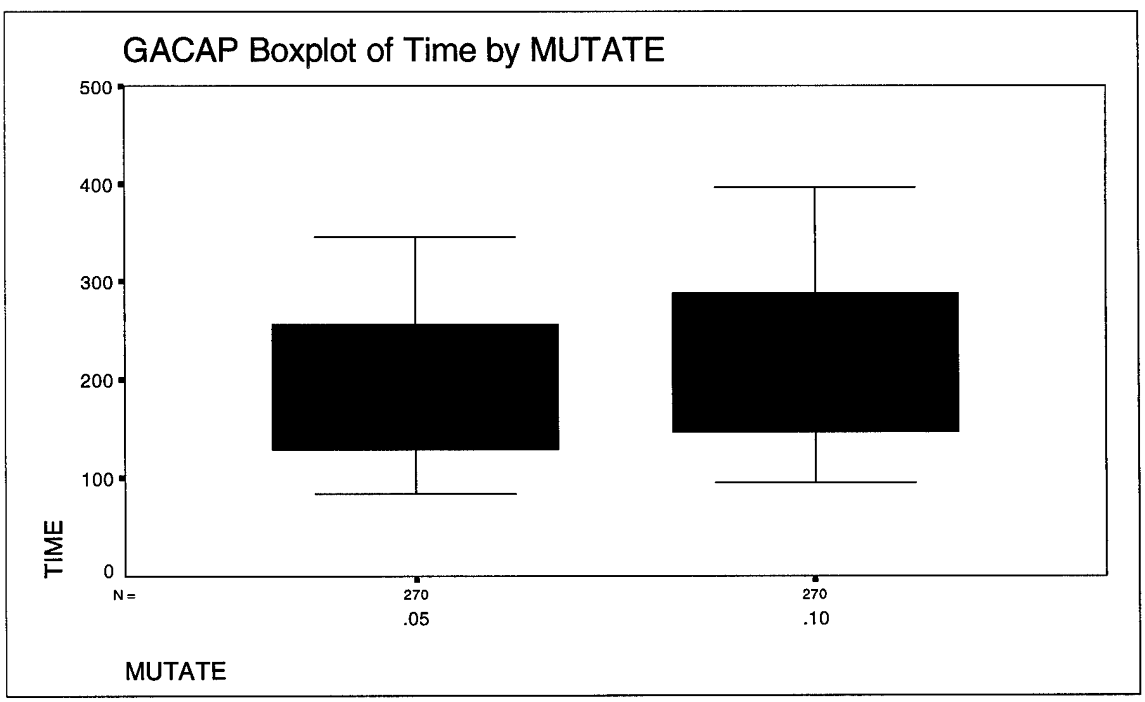

Figure 79. GACAP Boxplot of Time by MUTATE

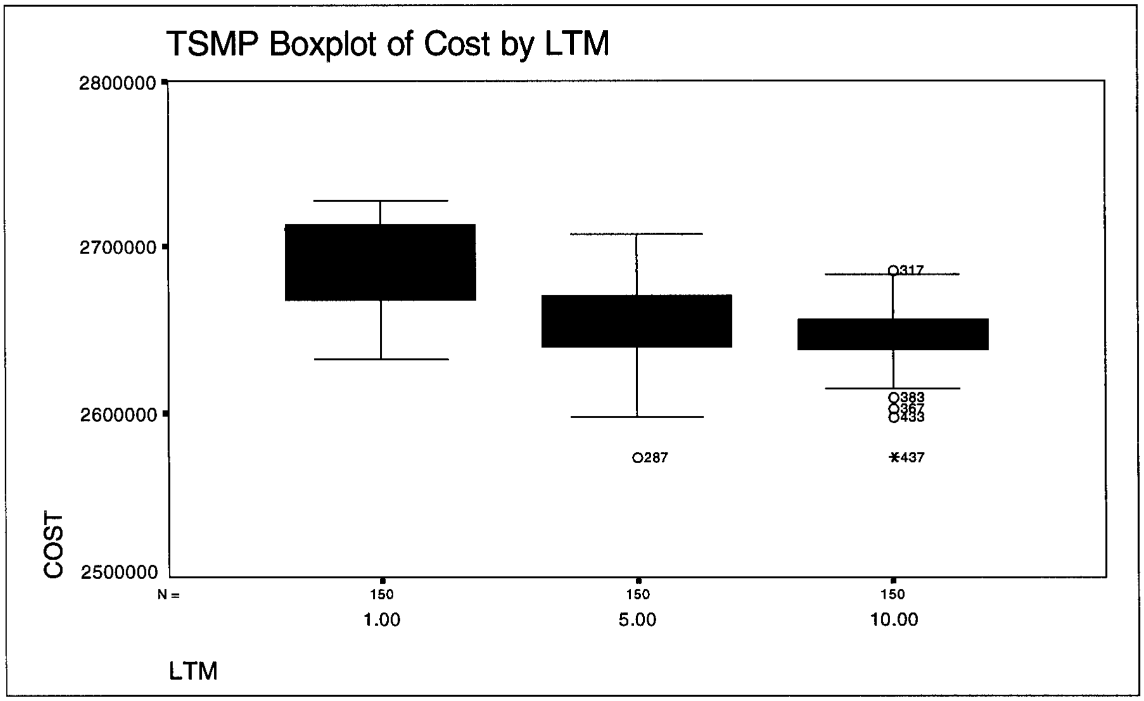

Figure 80. TSMP Boxplot of Cost by LTM

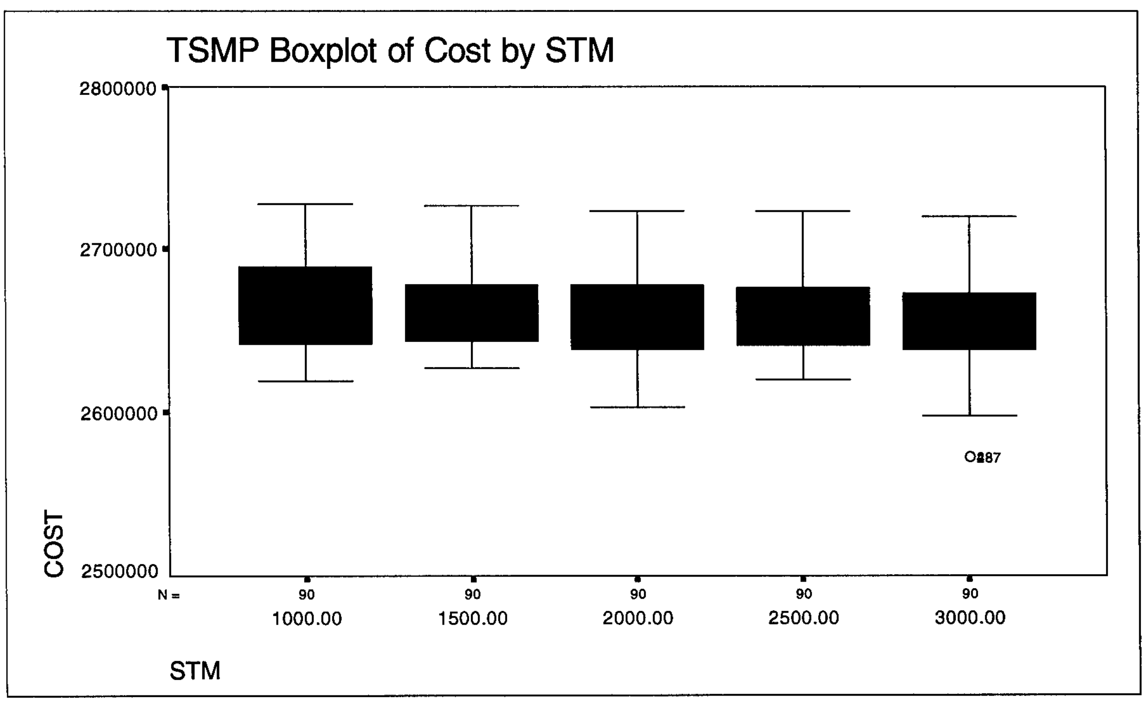

Figure 81. TSMP Boxplot of Cost by STM

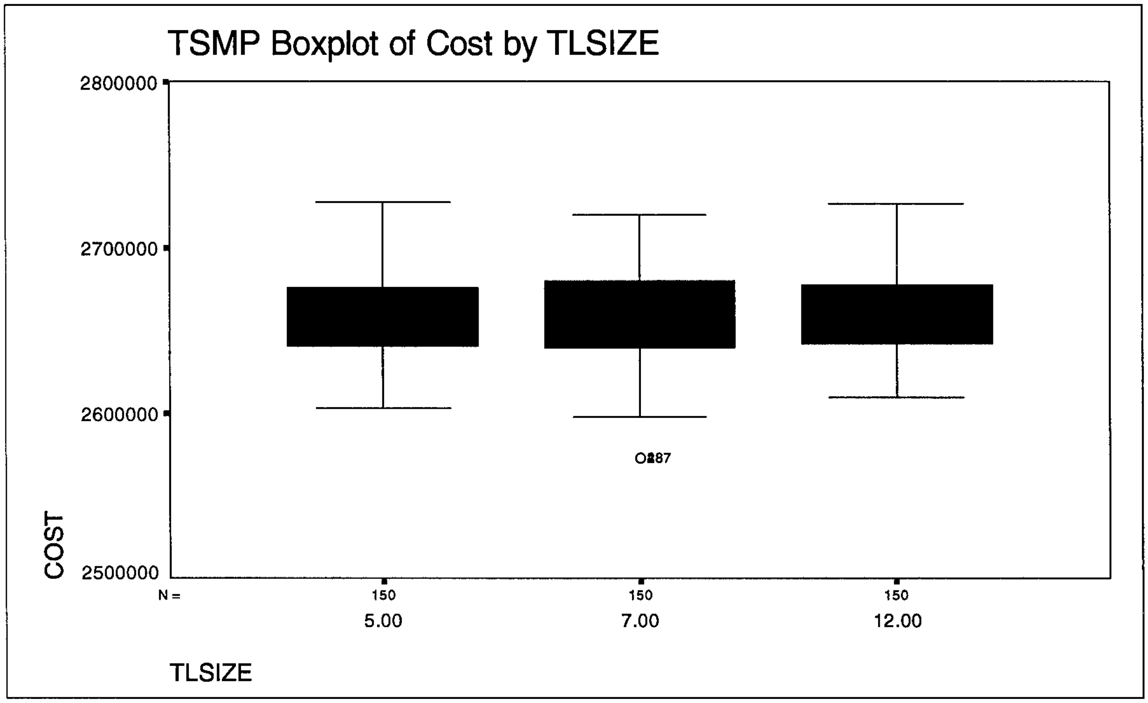

Figure 82. TSMP Boxplot of Cost by TLSIZE

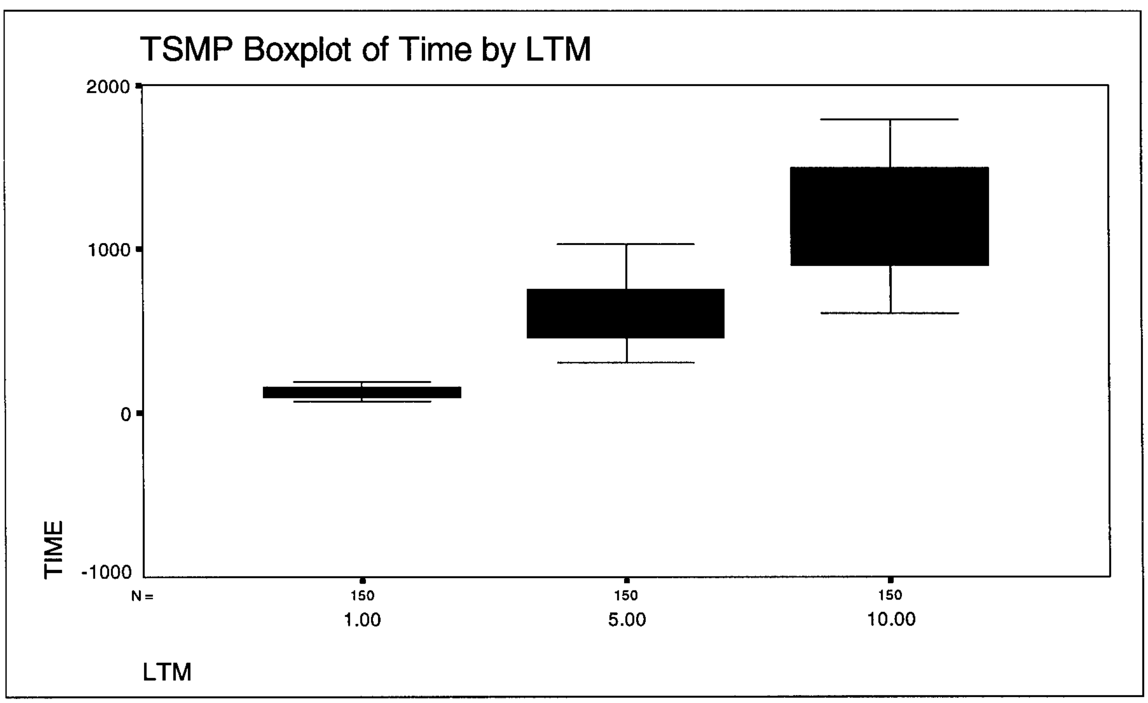

Figure 83. TSMP Boxplot of Time by LTM

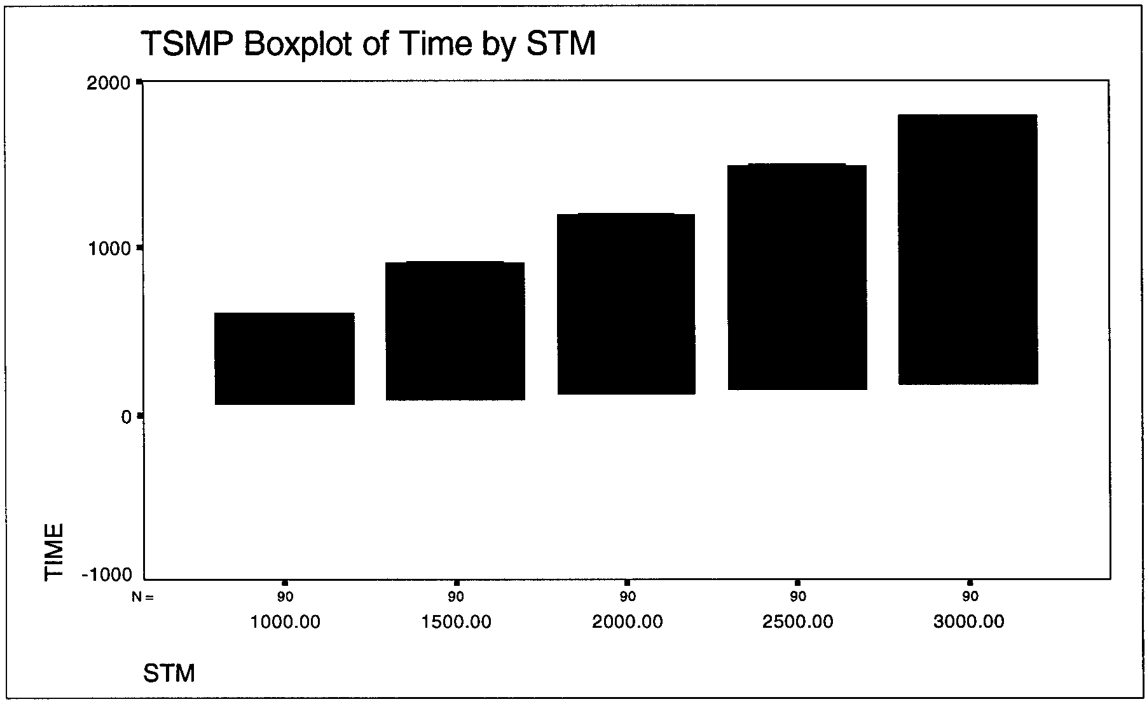

Figure 84. TSMP Boxplot of Time by STM

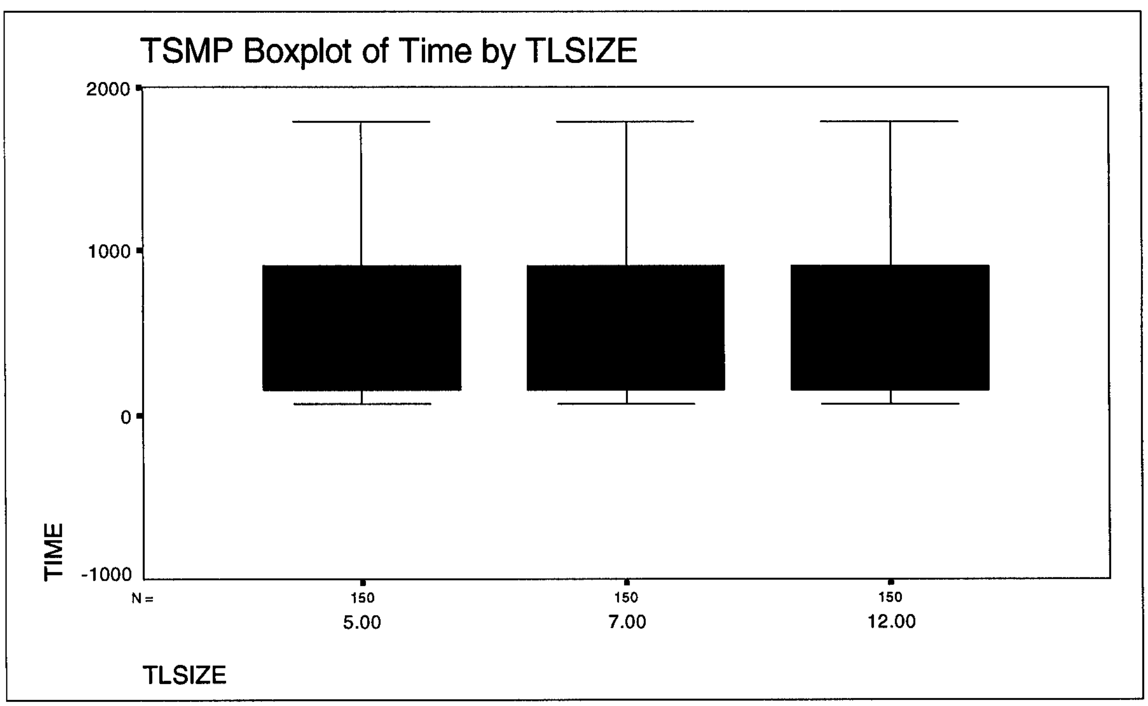

Figure 85. TSMP Boxplot of Time by TLSIZE

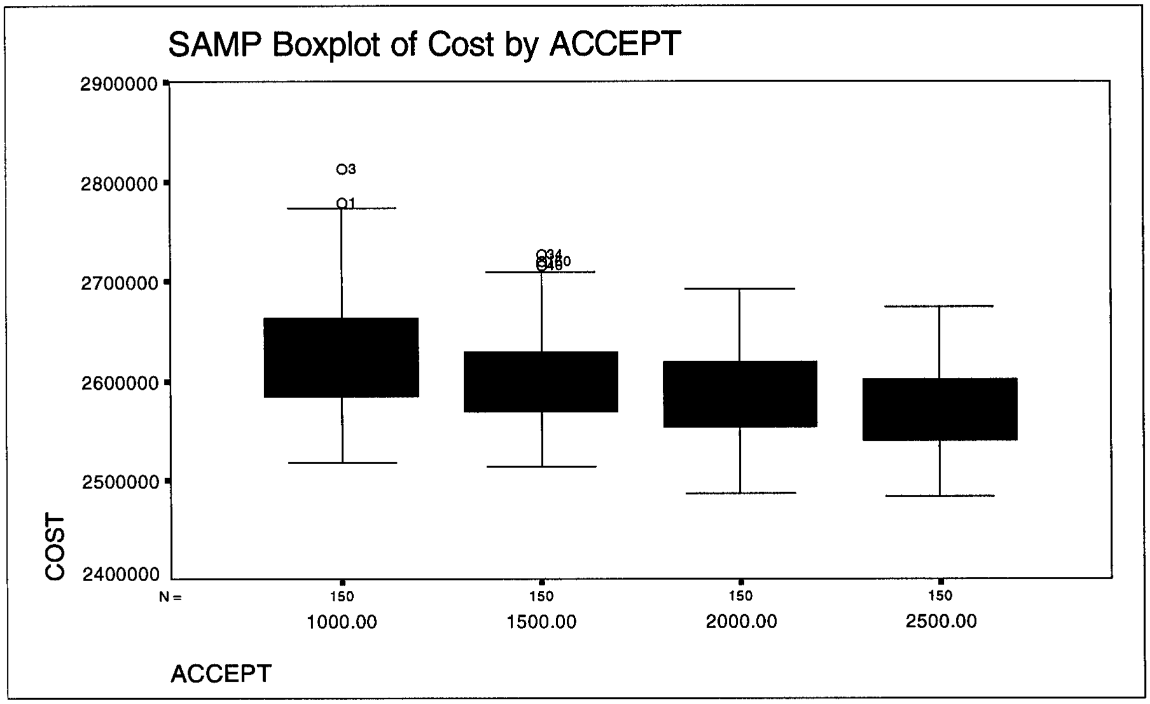

Figure 86. SAMP Boxplot of Cost by ACCEPT

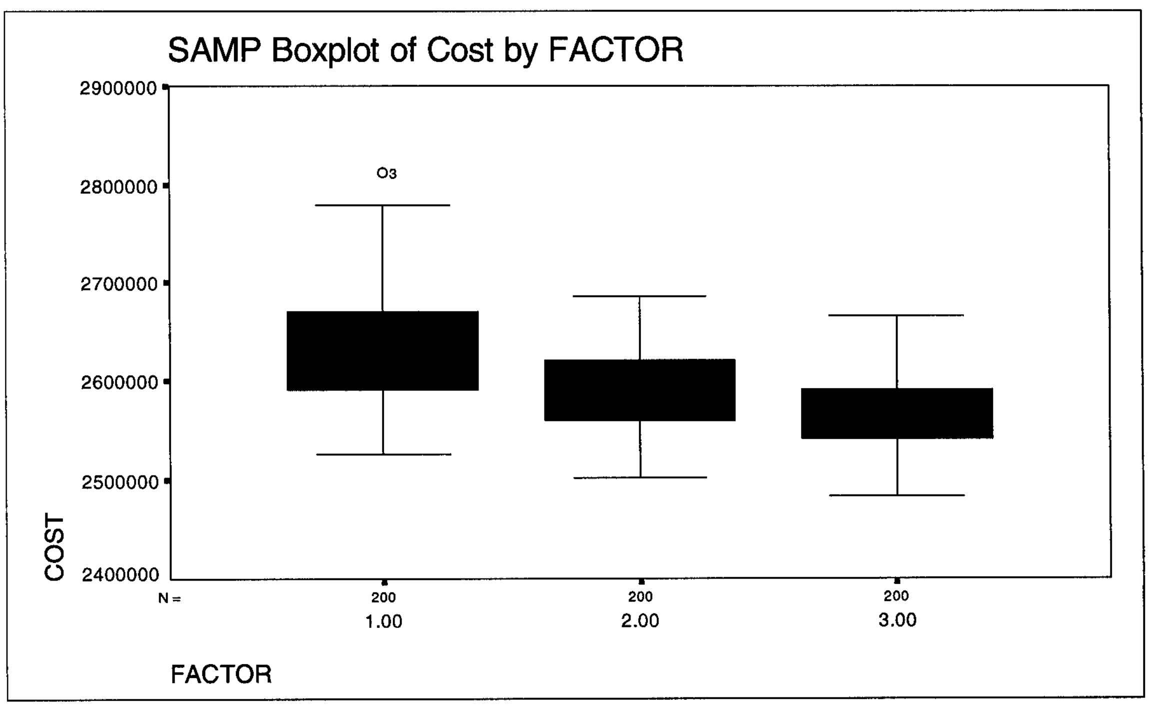

Figure 87. SAMP Boxplot of Cost by FACTOR

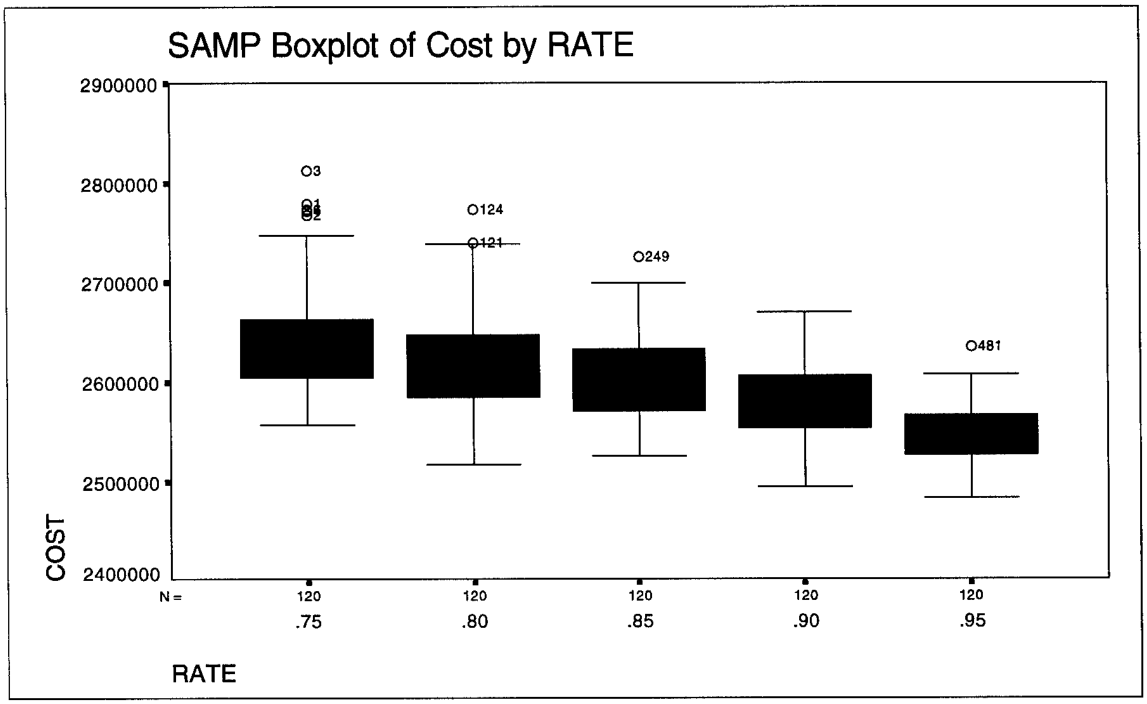

Figure 88. SAMP Boxplot of Cost by RATE

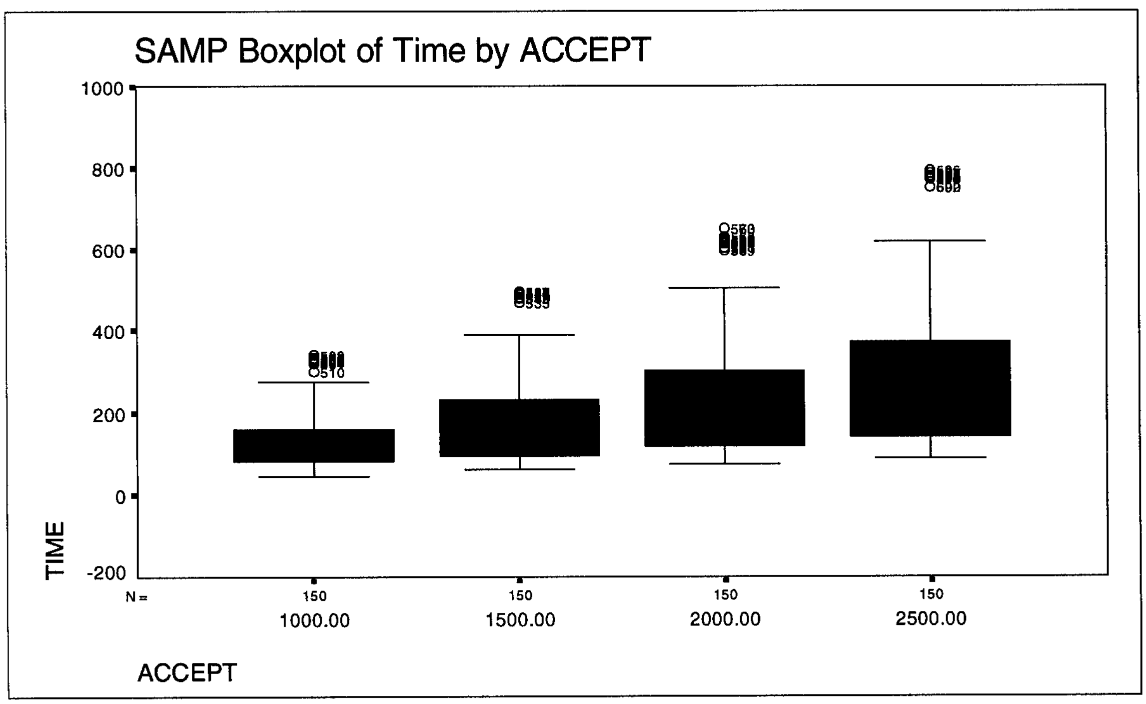

Figure 89. SAMP Boxplot of Time by ACCEPT

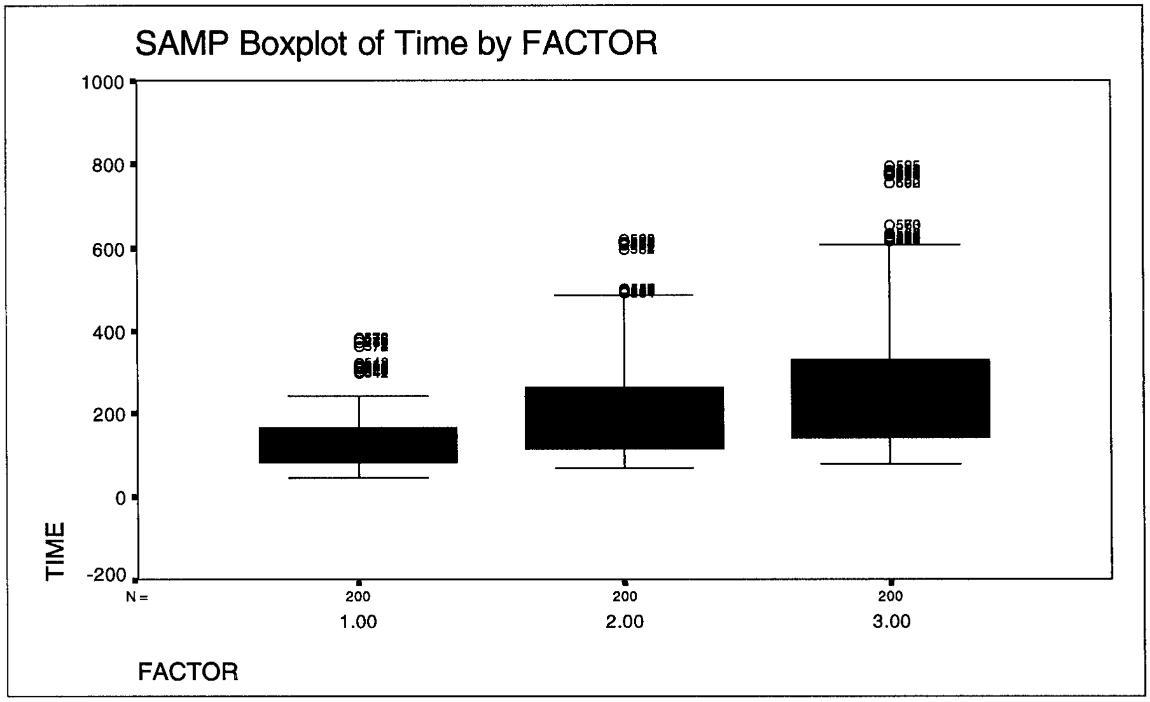

Figure 90. SAMP Boxplot of Time by FACTOR

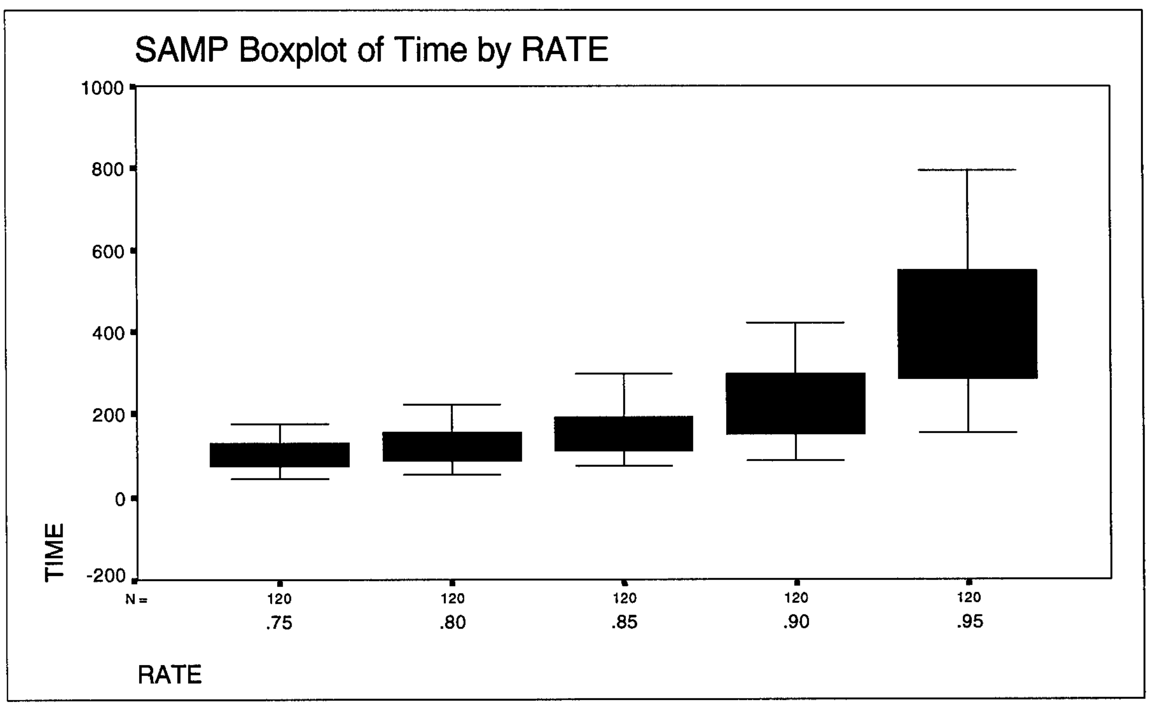

Figure 91. SAMP Boxplot of Time by RATE

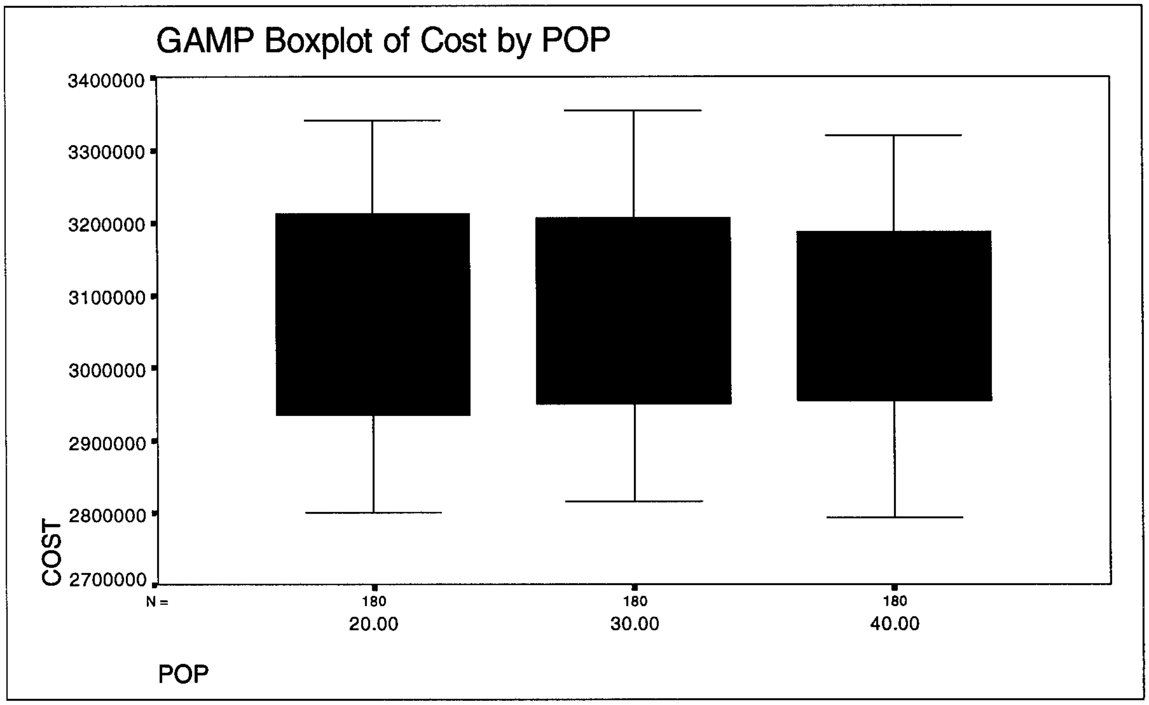

Figure 92. GAMP Boxplot of Cost by POP

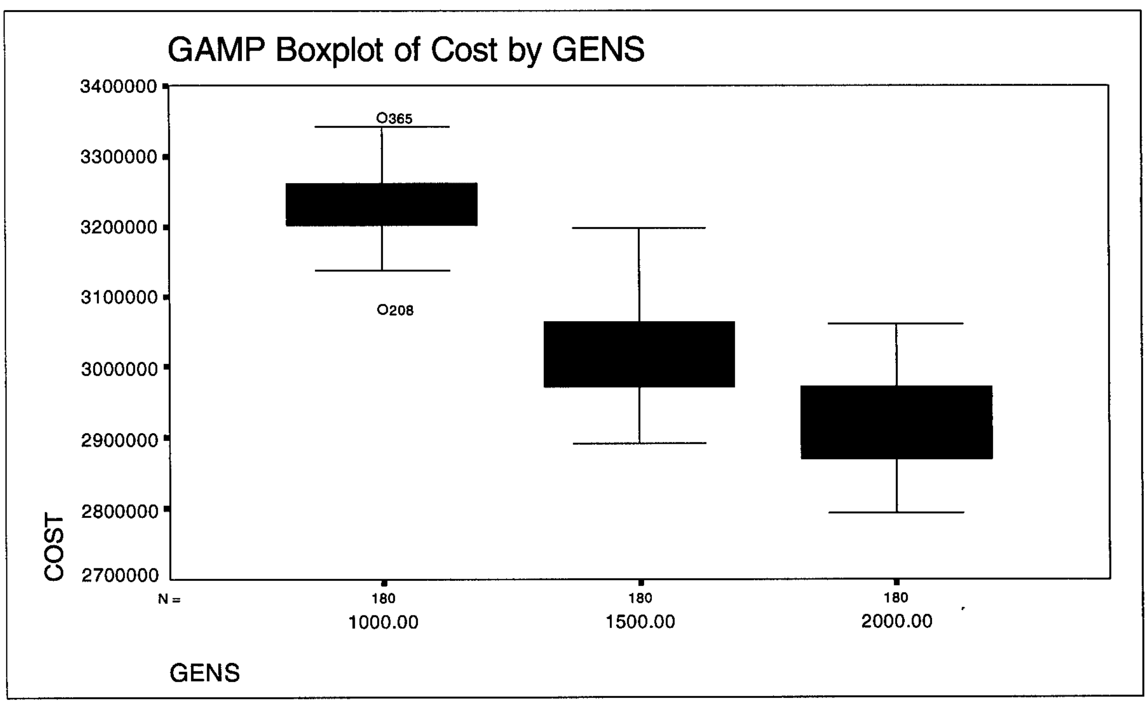

Figure 93. GAMP Boxplot of Cost by GENS

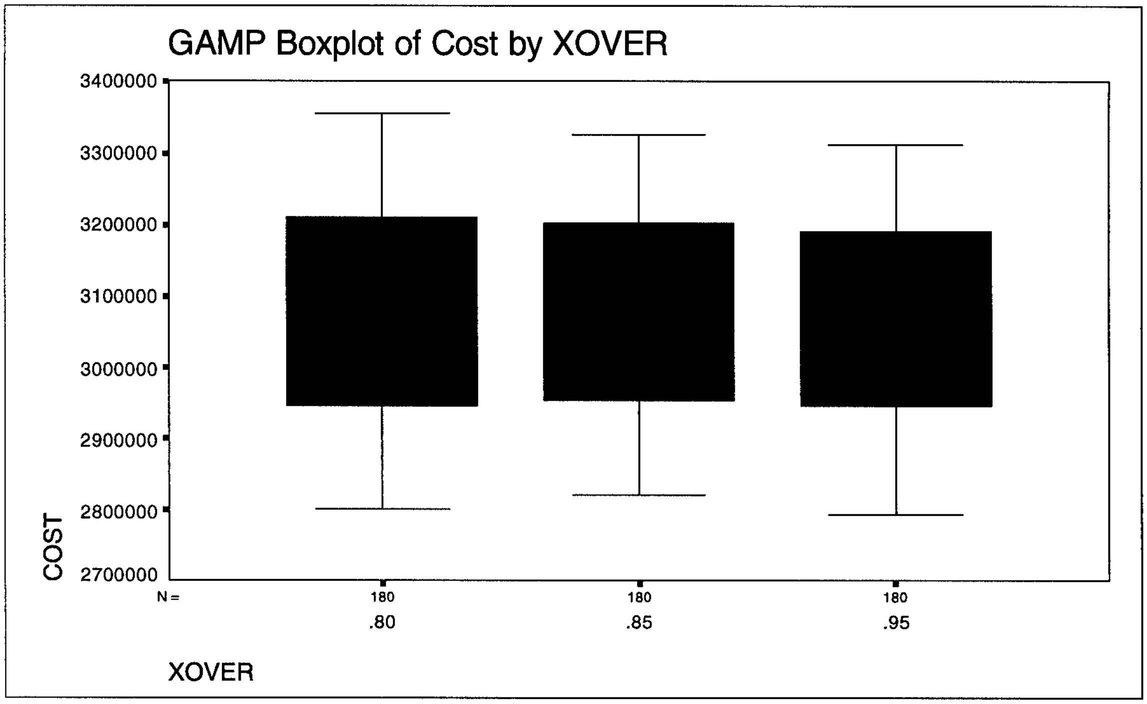

Figure 94. GAMP Boxplot of Cost by XOVER

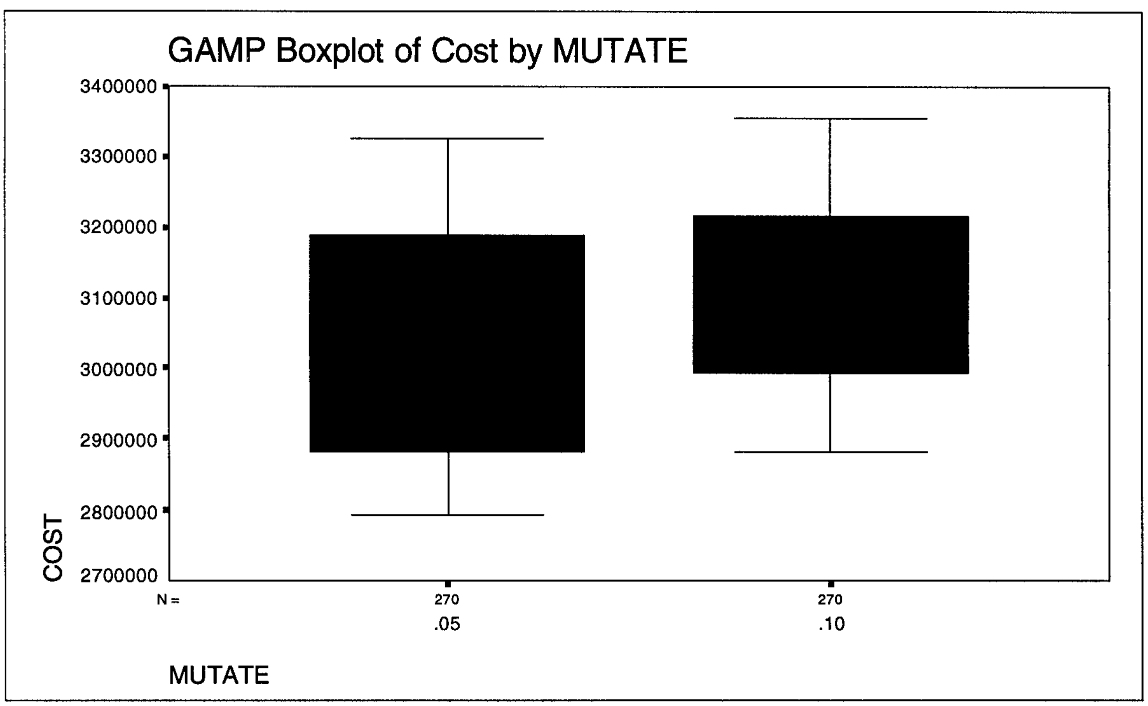

Figure 95. GAMP Boxplot of Cost by MUTATE

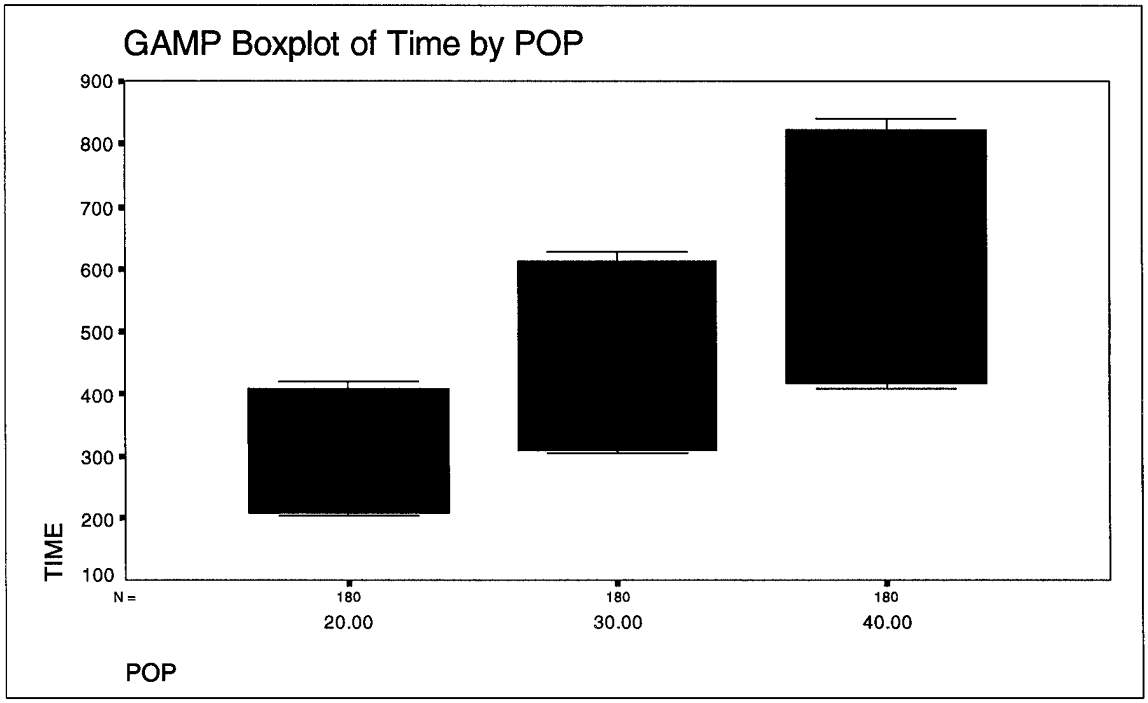

Figure 96. GAMP Boxplot of Time by POP

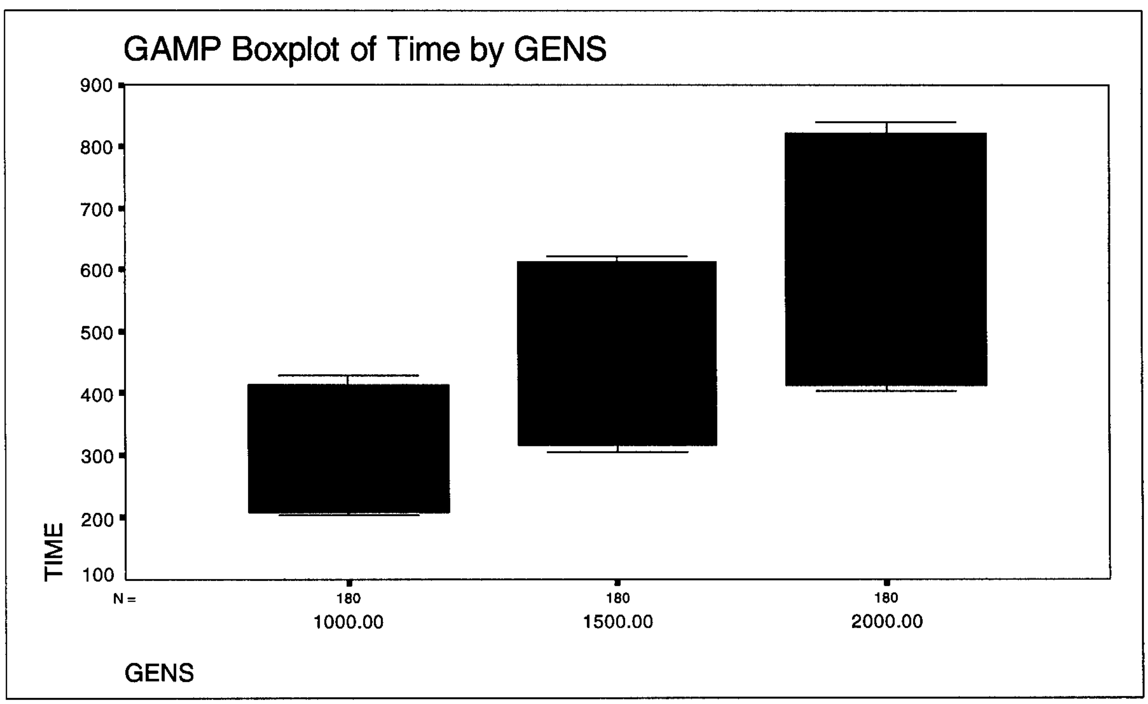

Figure 97. GAMP Boxplot of Time by GENS

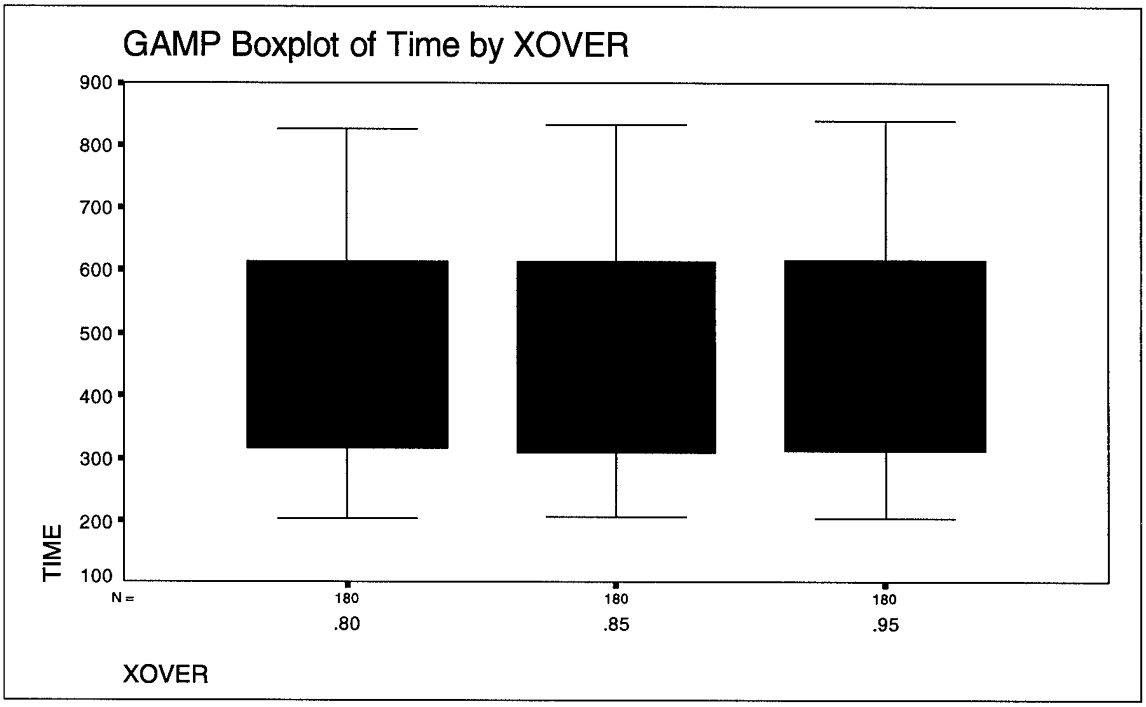

Figure 98. GAMP Boxplot of Time by XOVER

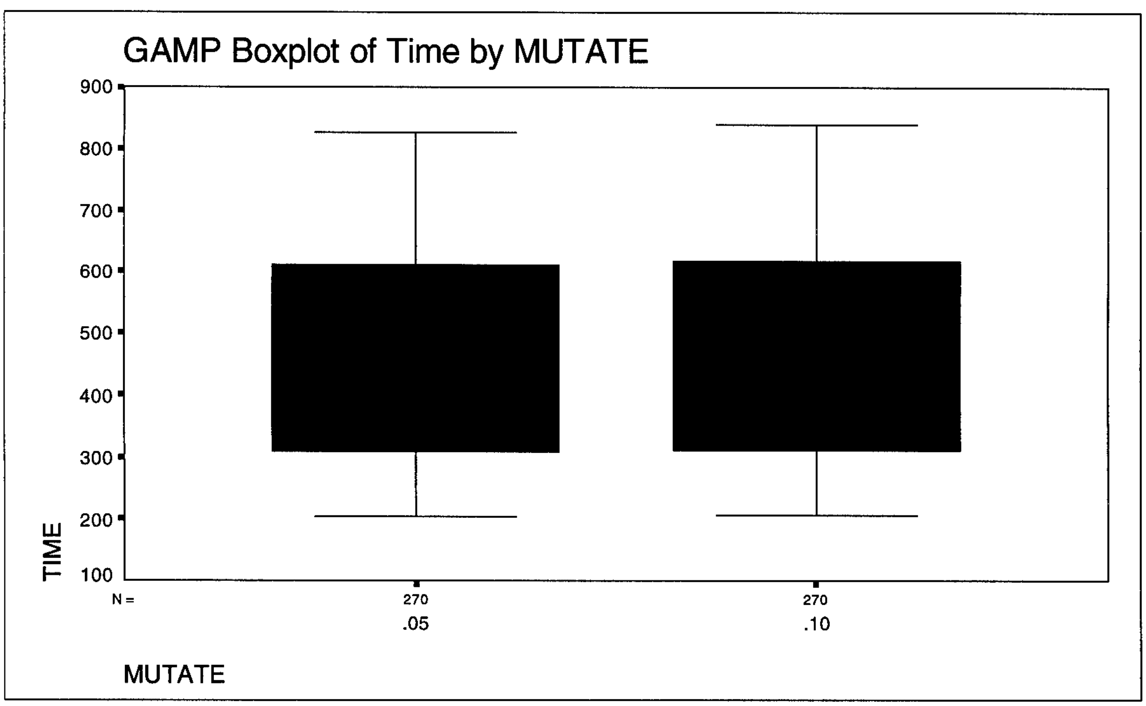

Figure 99. GAMP Boxplot of Time by MUTATE

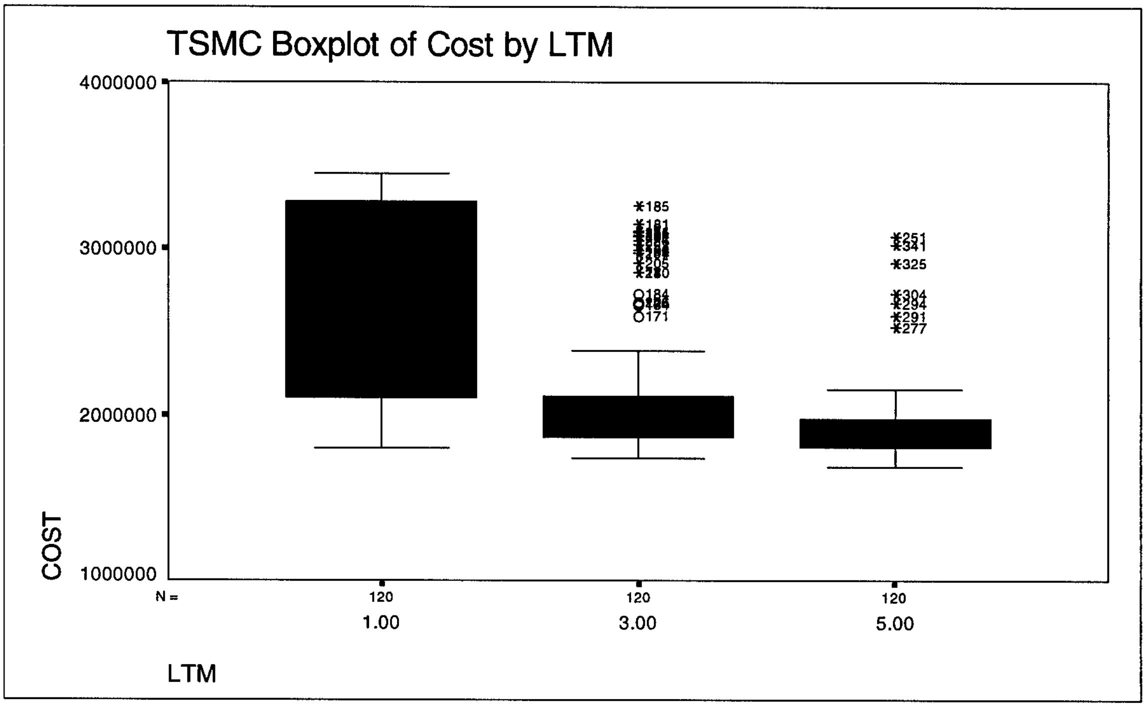

Figure 100. TSMC Boxplot of Cost by LTM

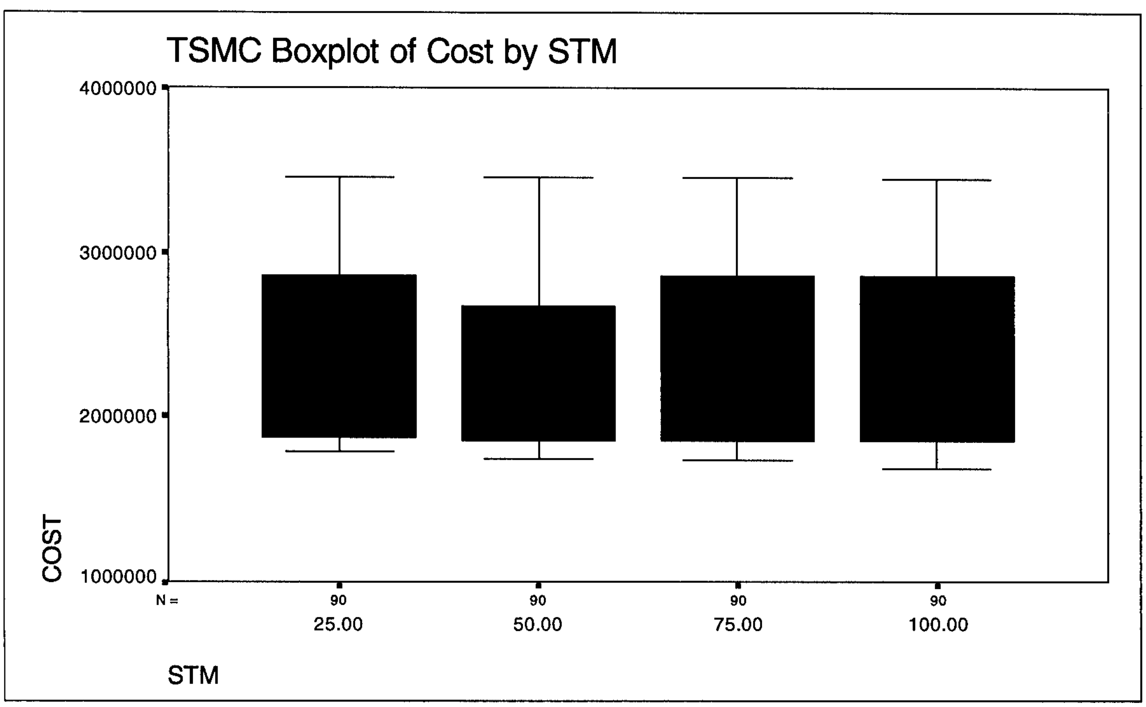

Figure 101. TSMC Boxplot of Cost by STM

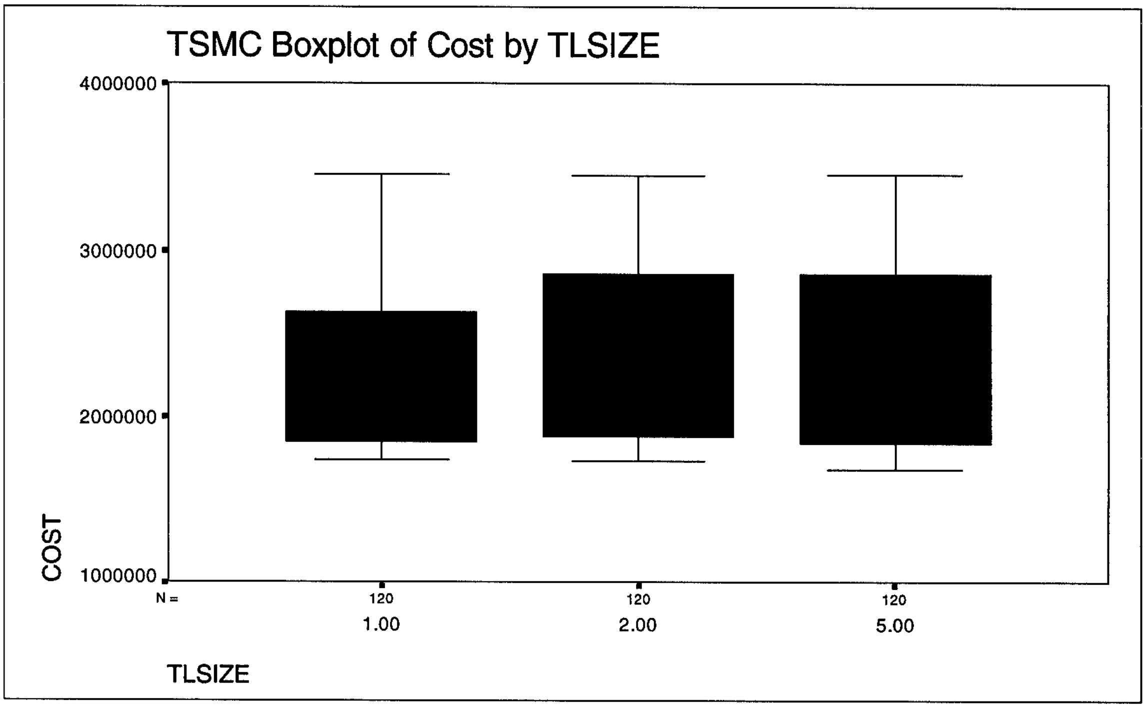

Figure 102. TSMC Boxplot of Cost by TLSIZE

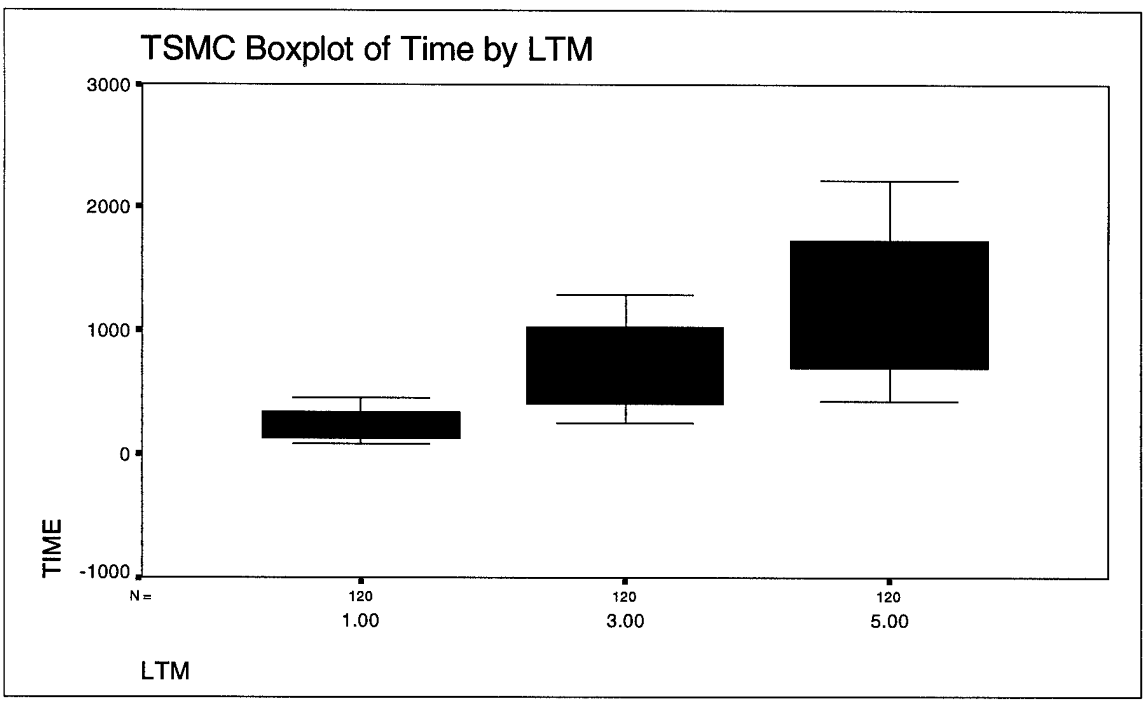

Figure 103. TSMC Boxplot of Time by LTM

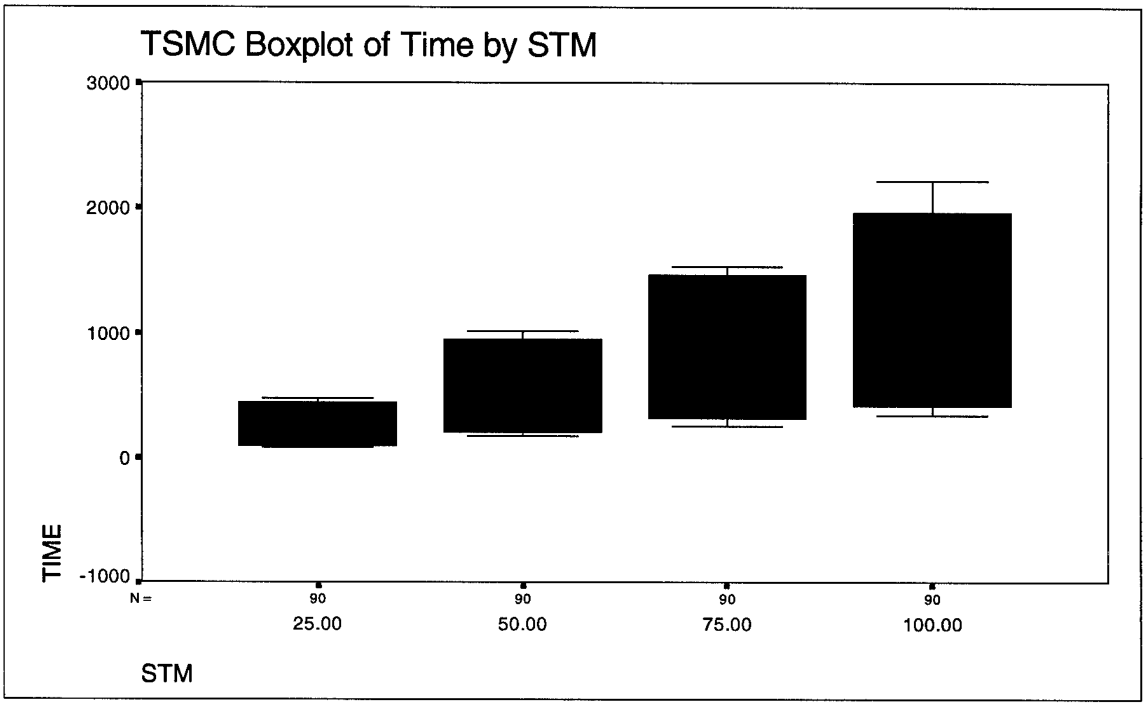

Figure 104. TSMC Boxplot of Time by STM

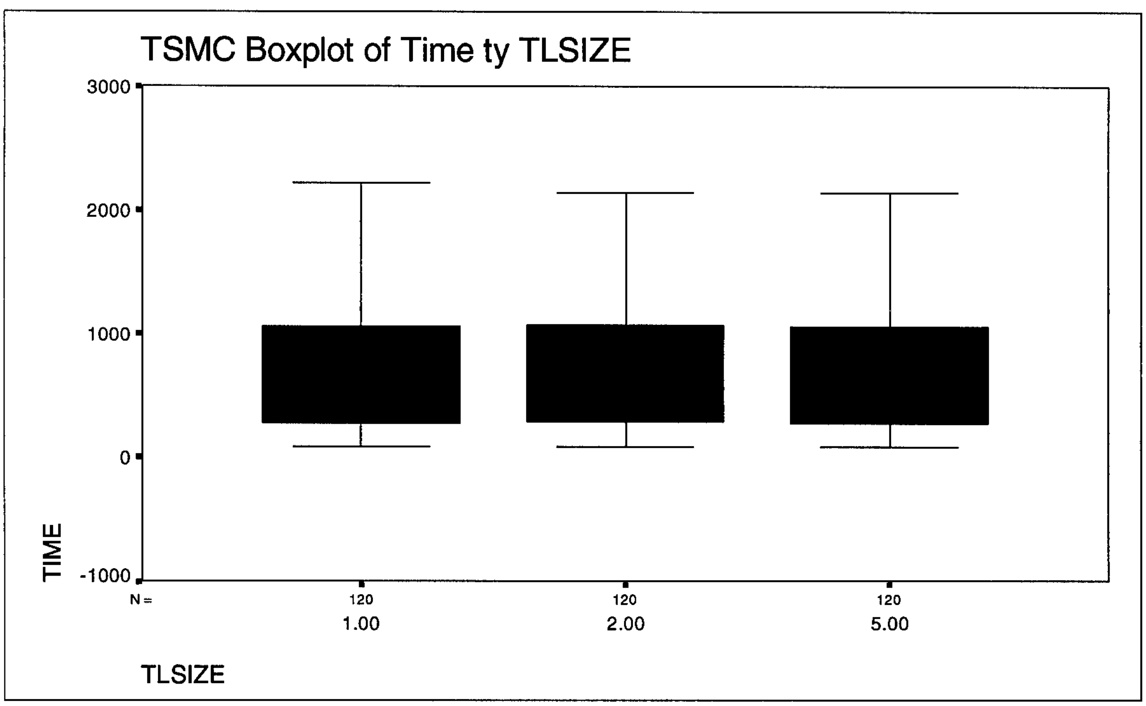

Figure 105. TSMC Boxplot of Time by TLSIZE

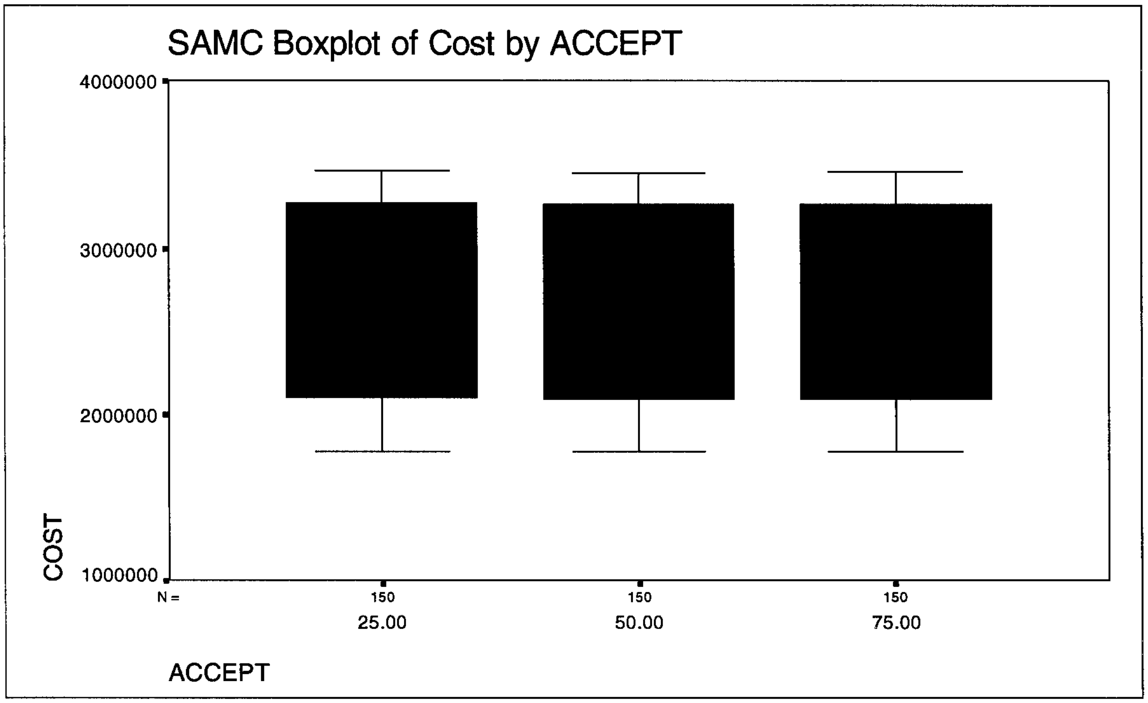

Figure 106. SAMC Boxplot of Cost by ACCEPT

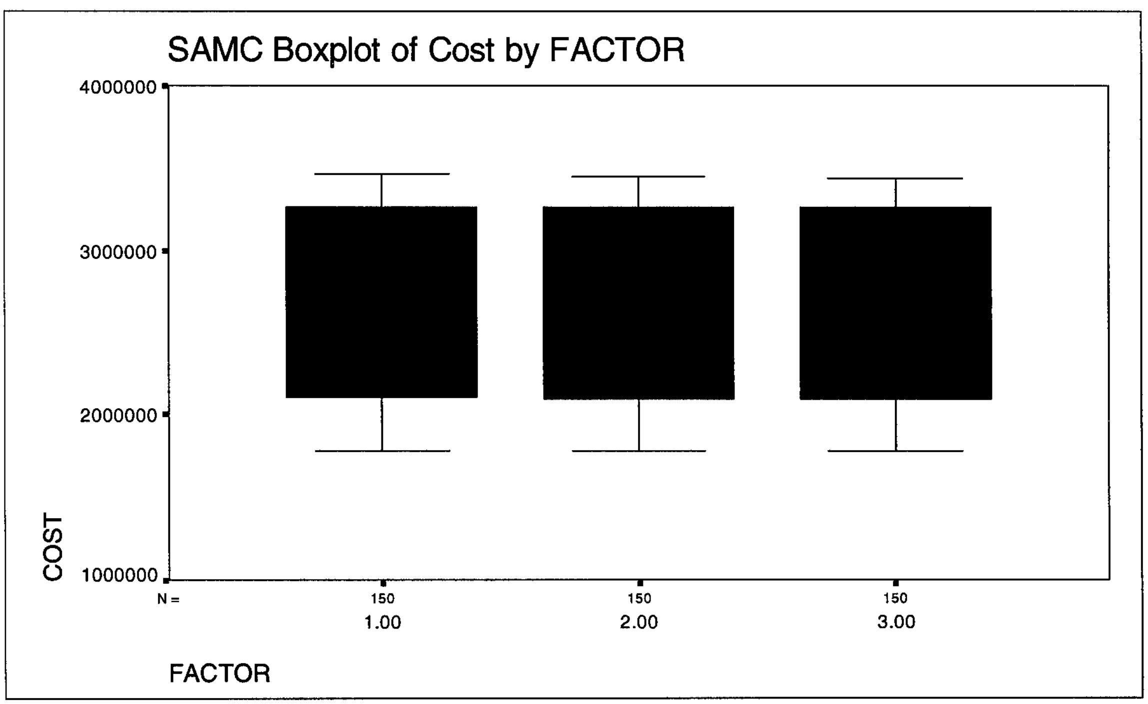

Figure 107. SAMC Boxplot of Cost by FACTOR

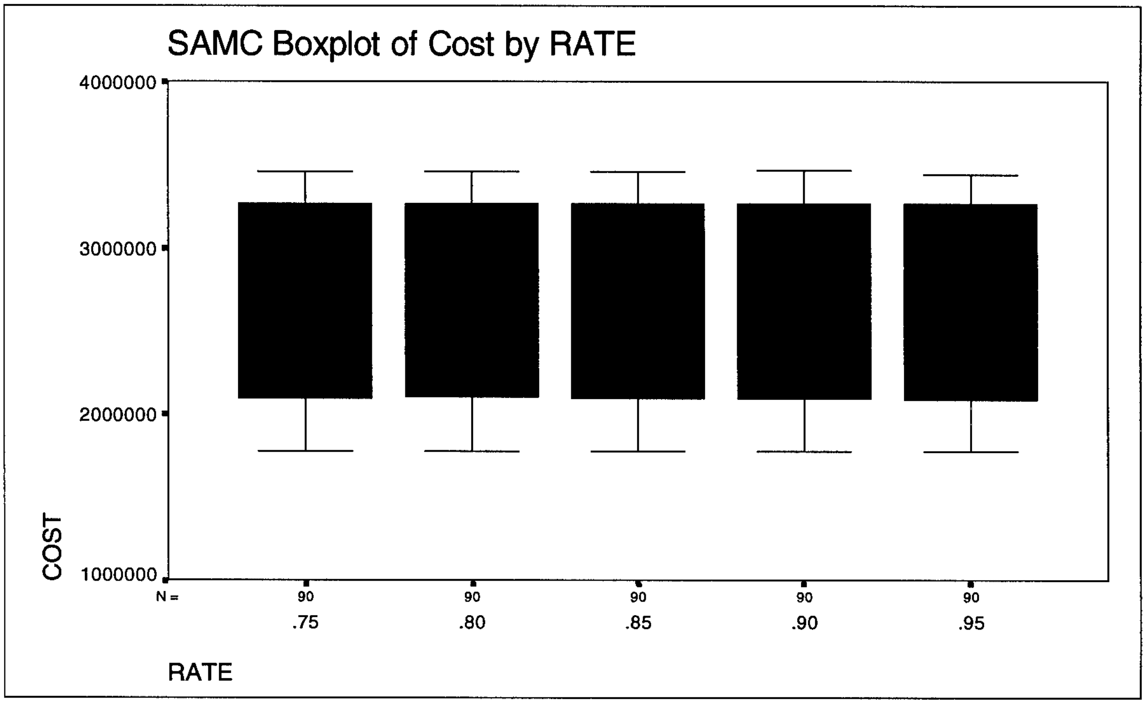

Figure 108. SAMC Boxplot of Cost by RATE

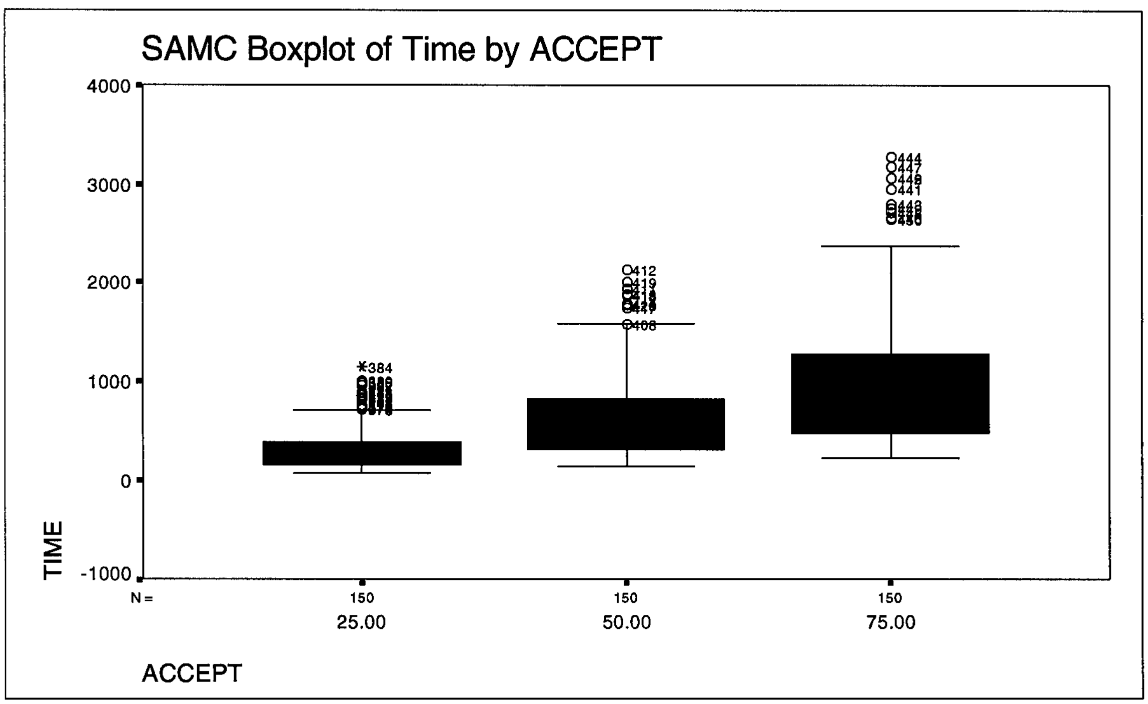

Figure 109. SAMC Boxplot of Time by ACCEPT

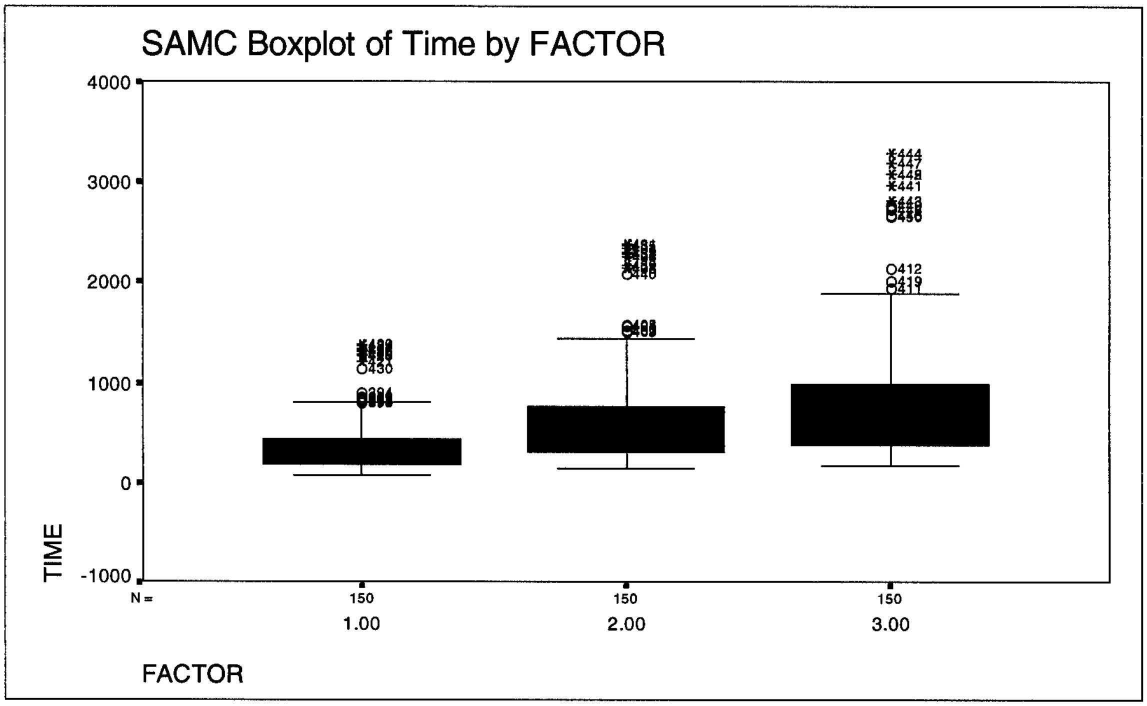

Figure 110. SAMC Boxplot of Time by FACTOR

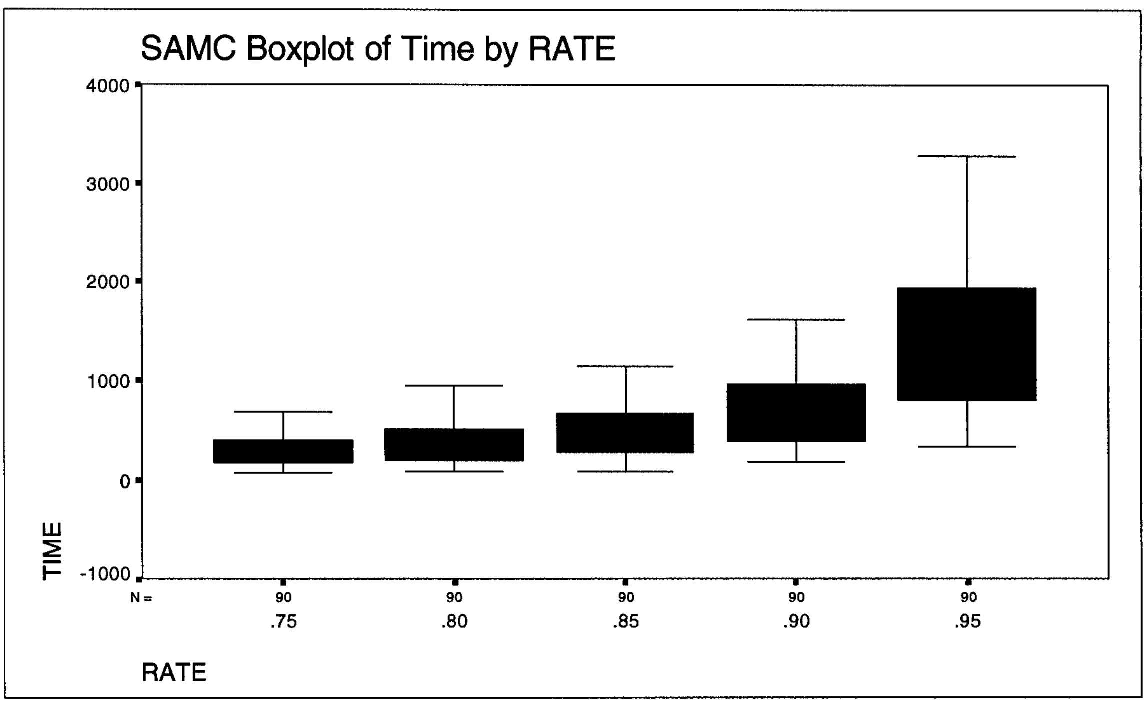

Figure 111. SAMC Boxplot of Time by RATE

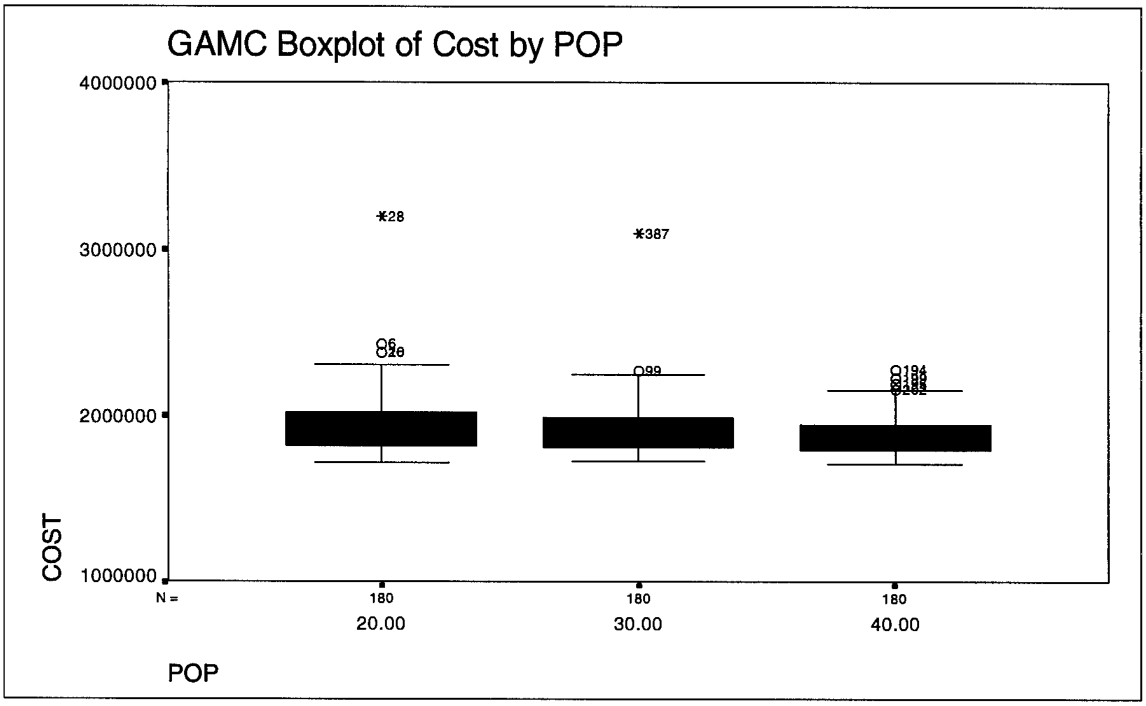

Figure 112. GAMC Boxplot of Cost by POP

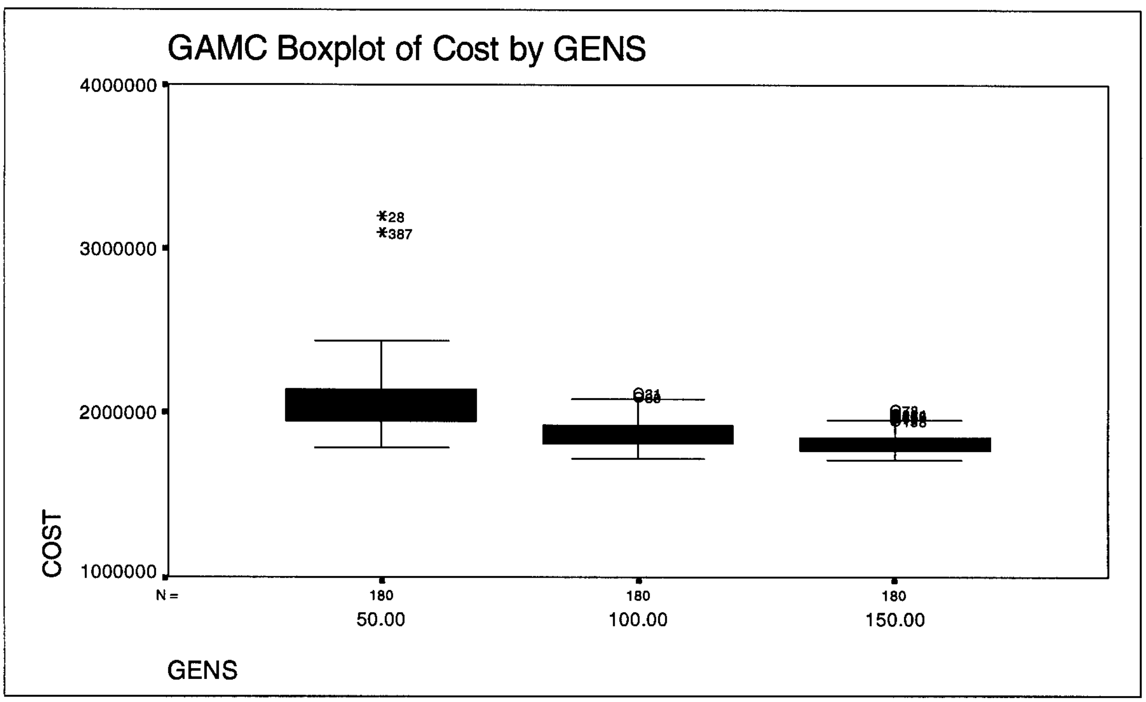

Figure 113. GAMC Boxplot of Cost by GENS

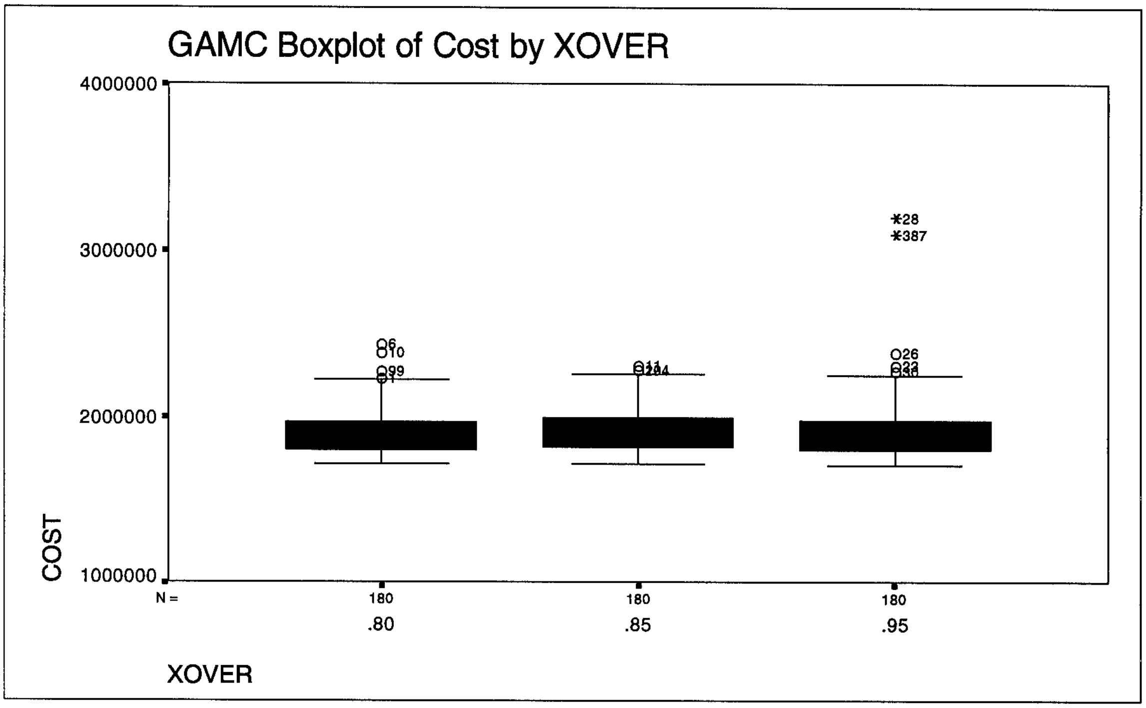

Figure 114. GAMC Boxplot of Cost by XOVER

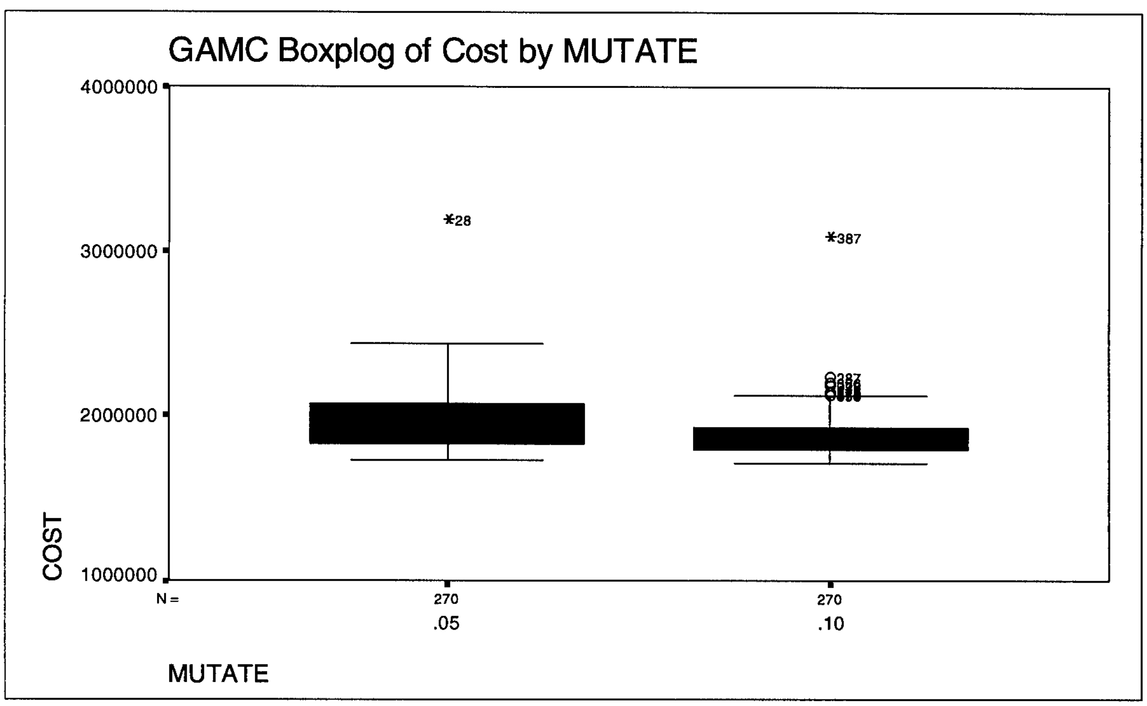

Figure 115. GAMC Boxplot of Cost by MUTATE

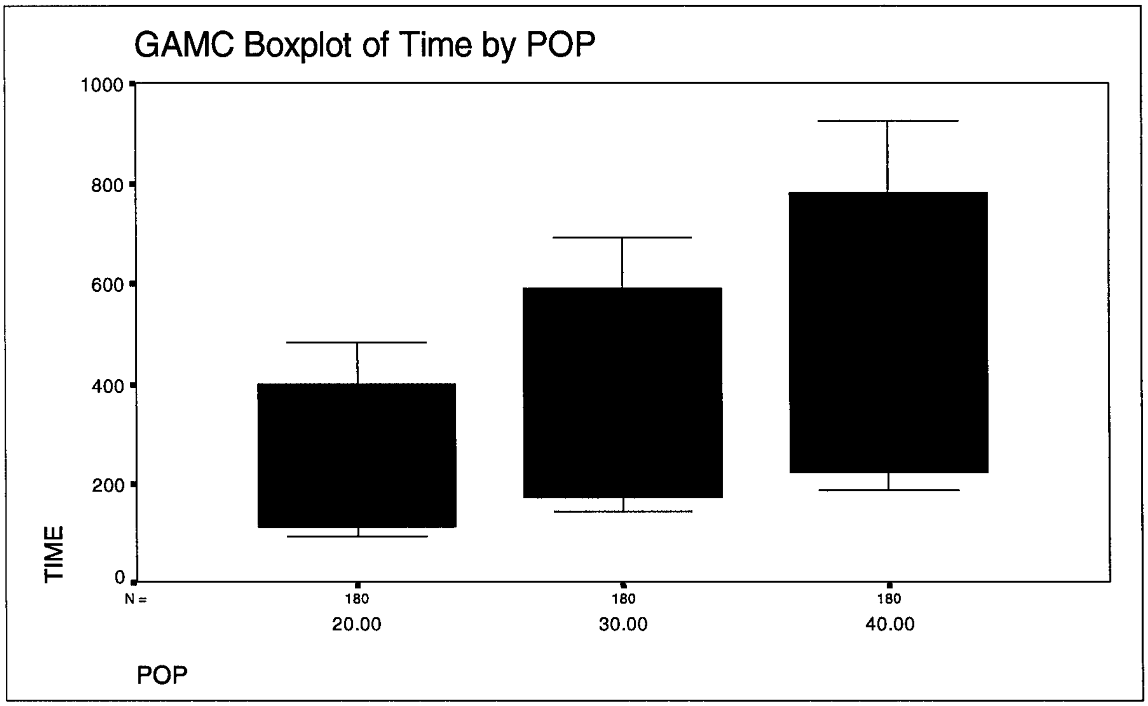

Figure 116. GAMC Boxplot of Time by POP

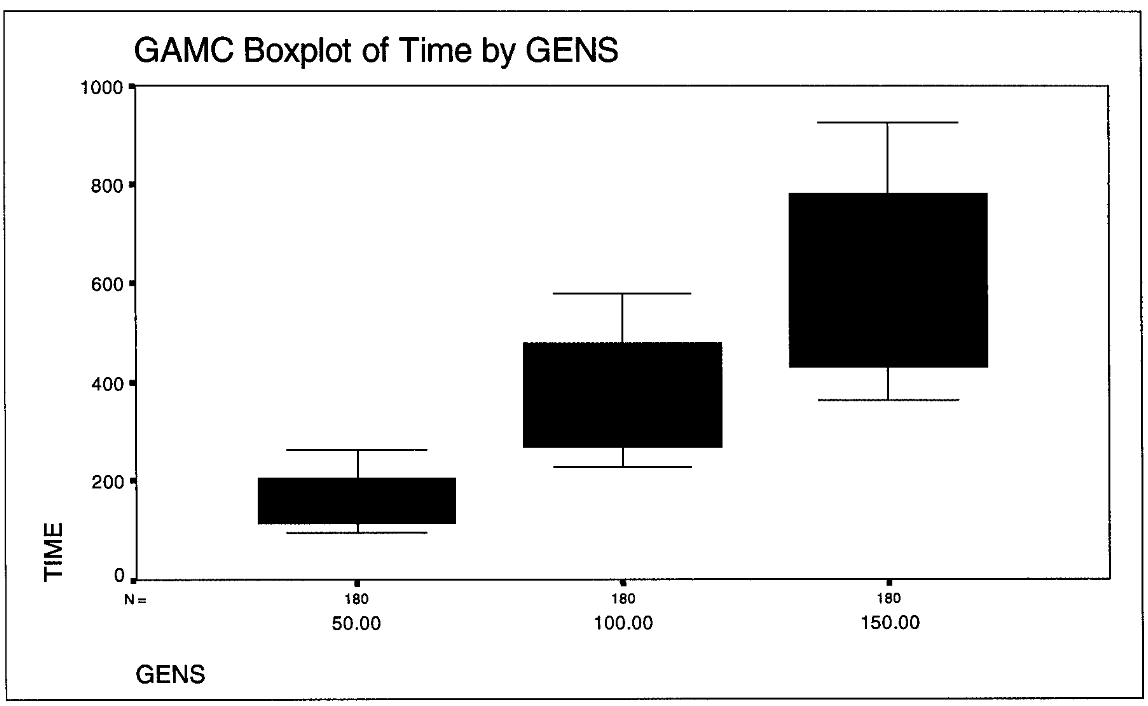

Figure 117. GAMC Boxplot of Time by GENS

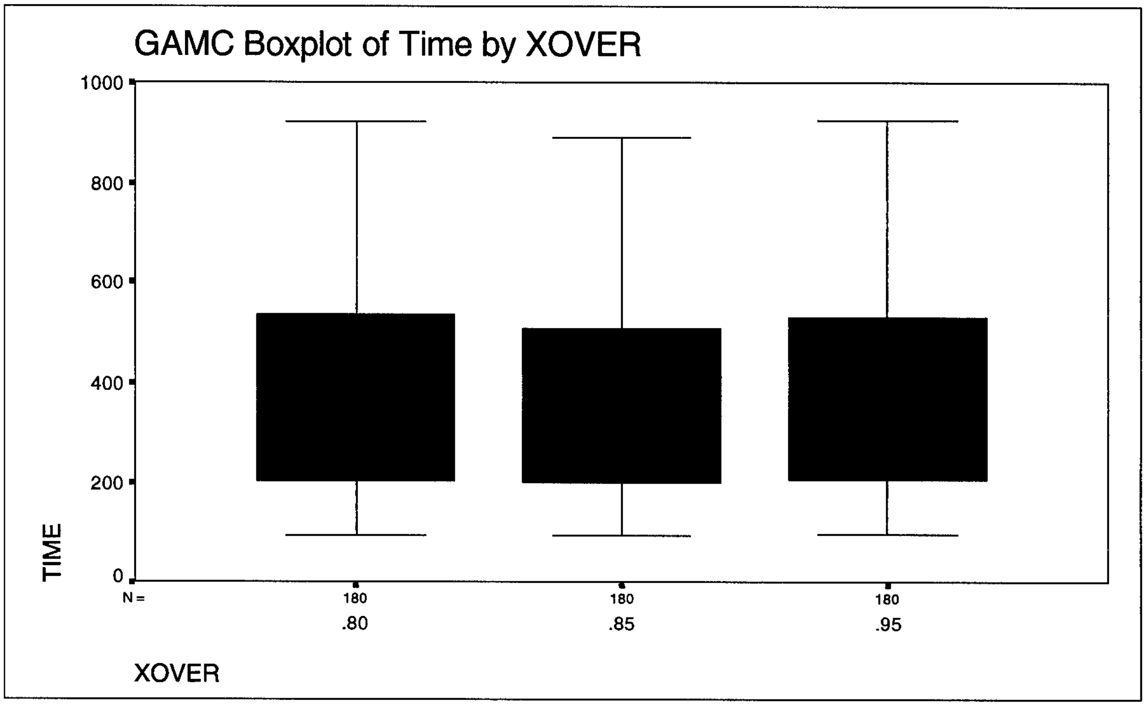

Figure 118. GAMC Boxplot of Time by XOVER

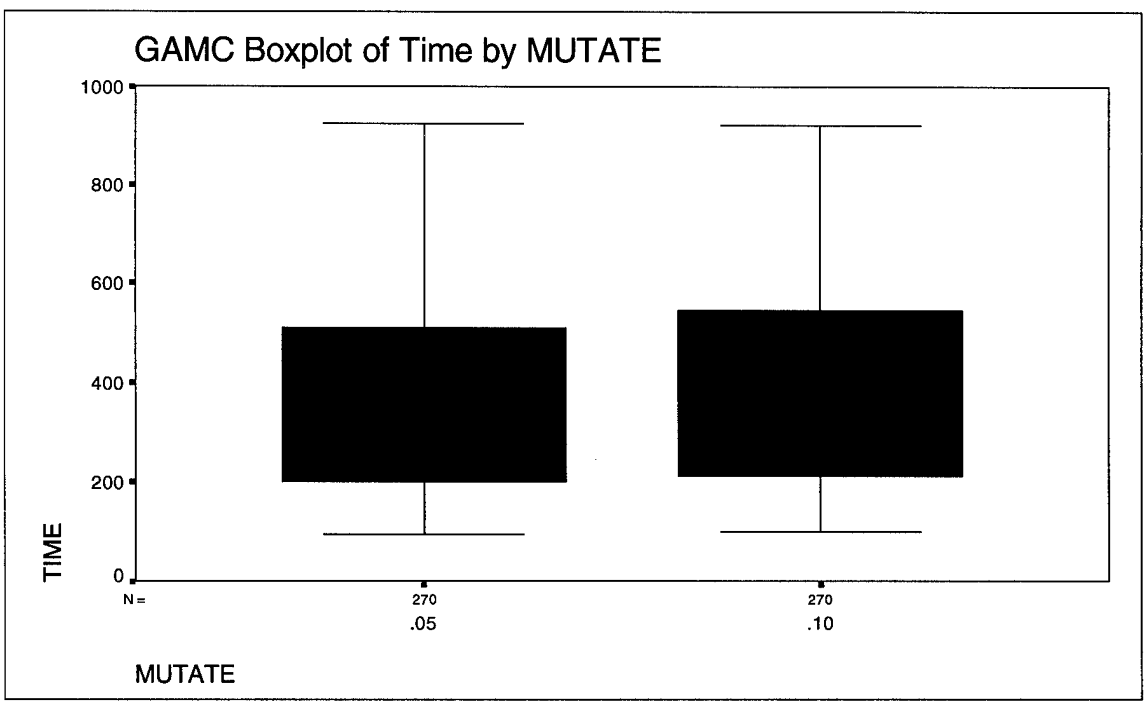

Figure 119. GAMC Boxplot of Time by MUTATE

APPENDIX E

TIME PERFORMANCE CHARTS FOR CFLP

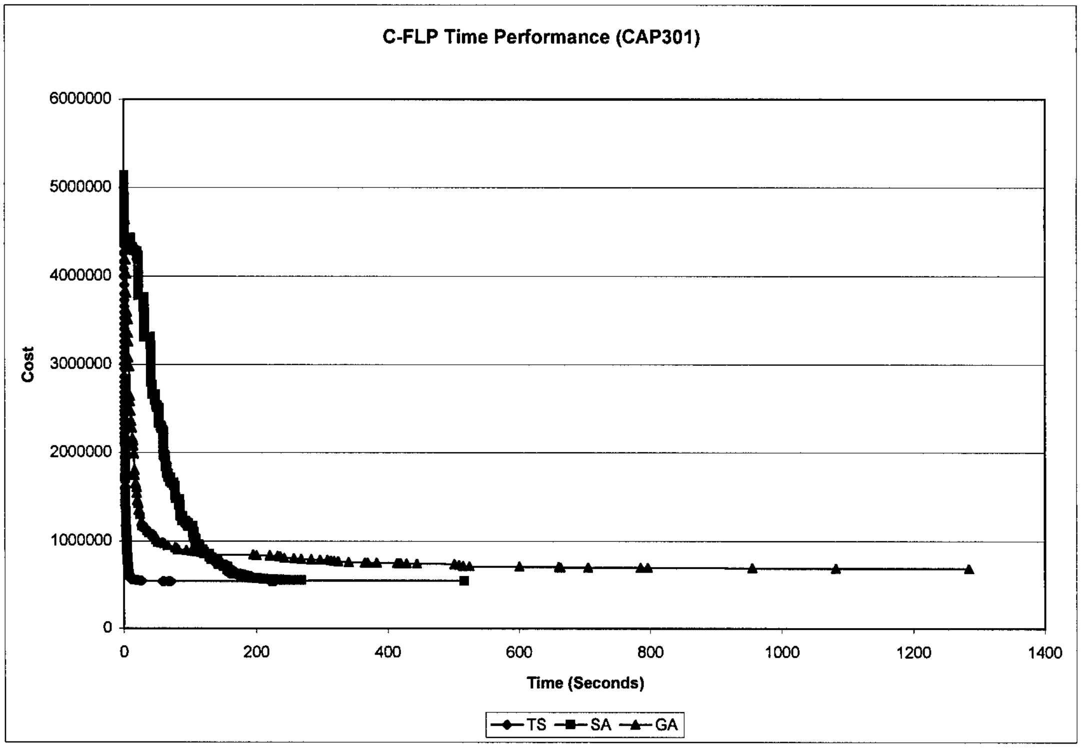

Figure 120. CFLP Time Performance (CAP301)

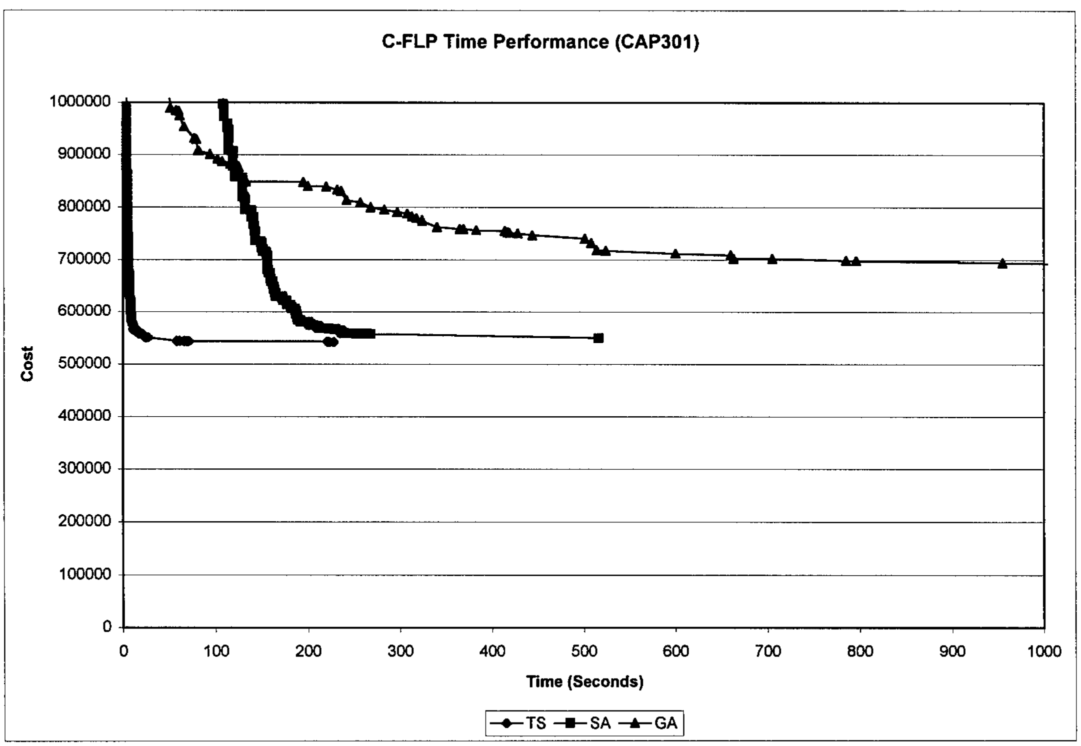

Figure 121. CFLP Time Performance Close-up (CAP301)

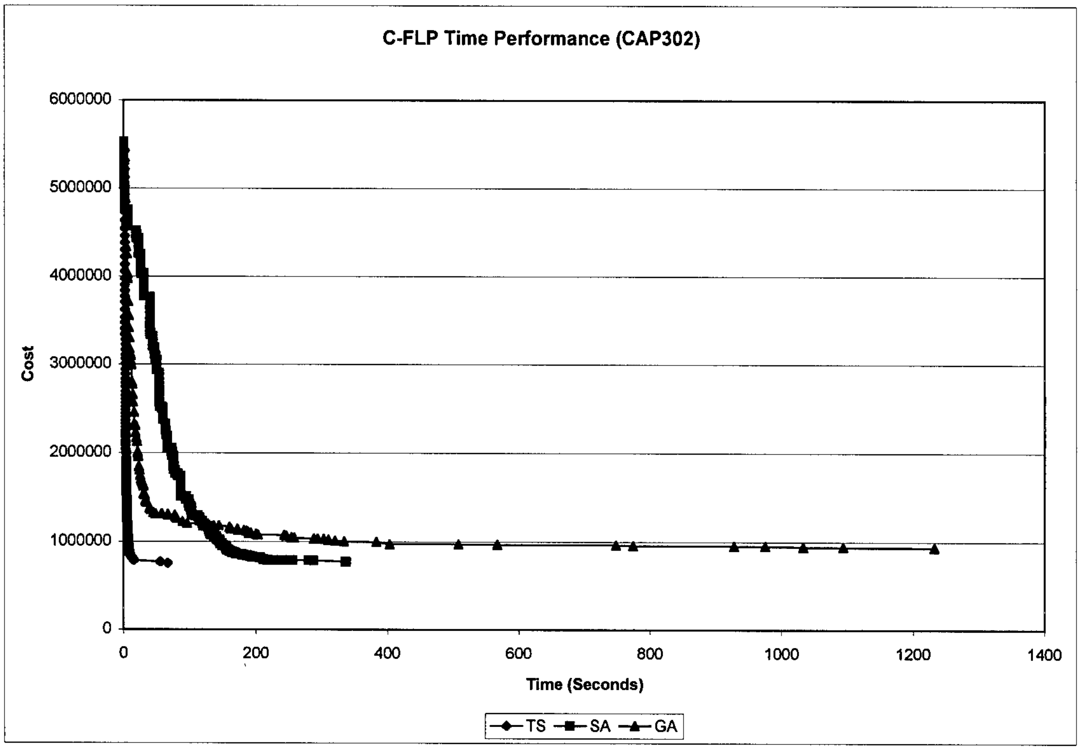

Figure 122. CFLP Time Performance (CAP302)

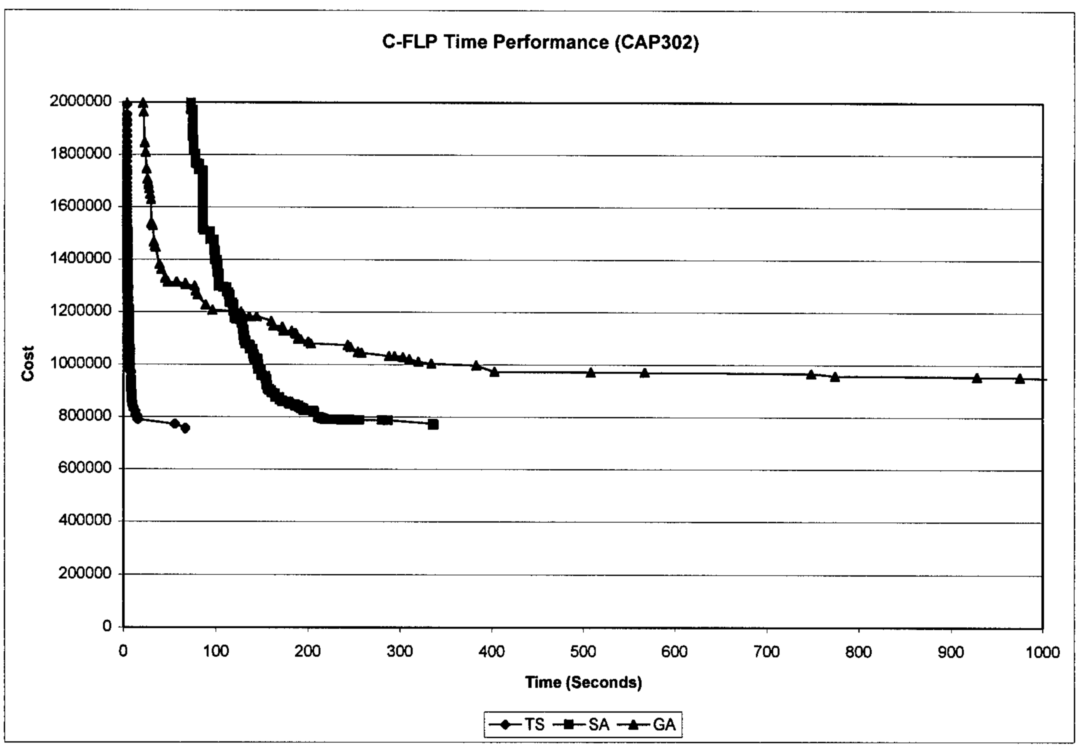

Figure 123. CFLP Time Performance Close-up (CAP302)

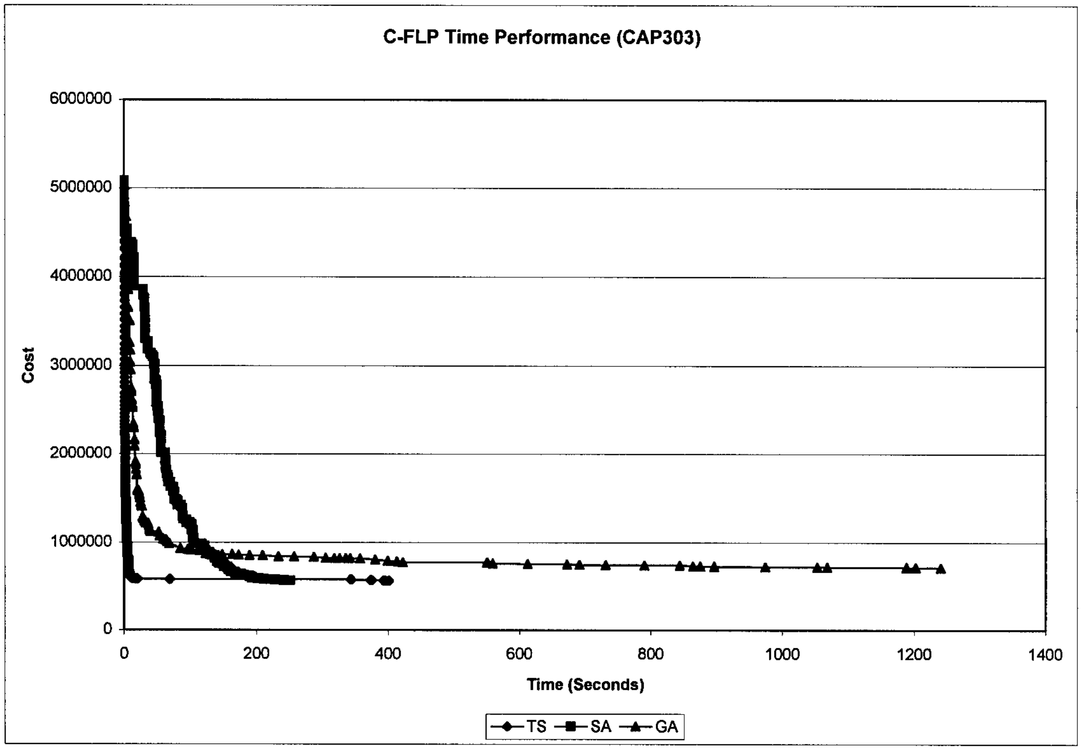

Figure 124. CFLP Time Performance (CAP303)

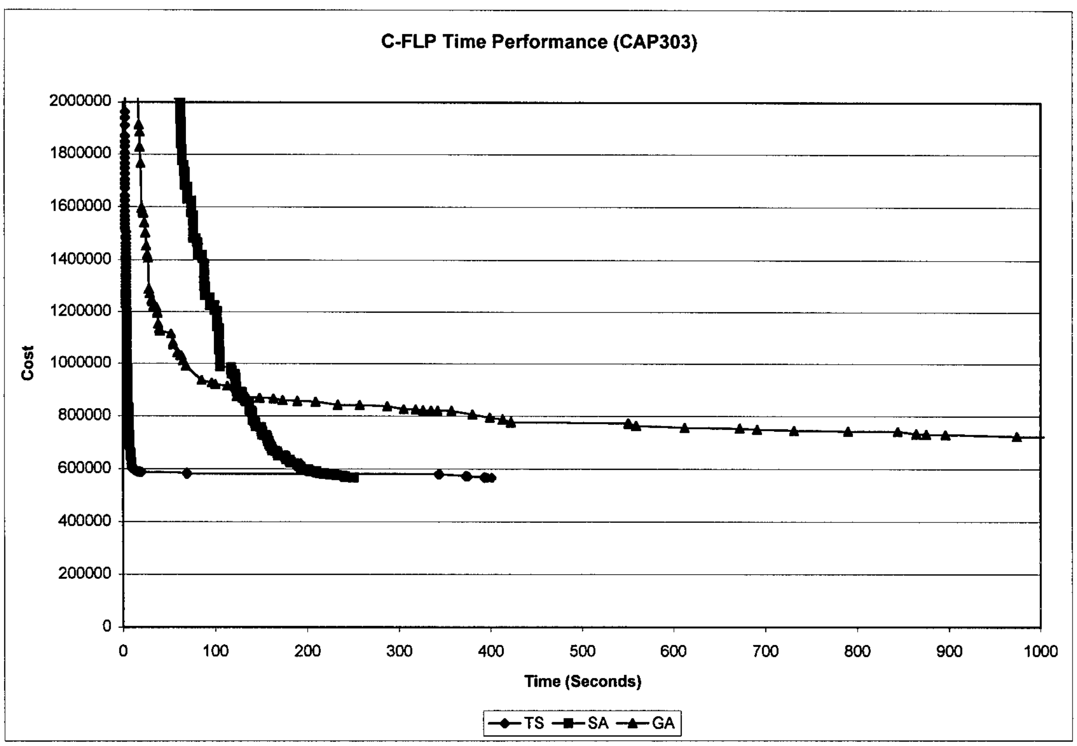

Figure 125. CFLP Time Performance Close-up (CAP303)

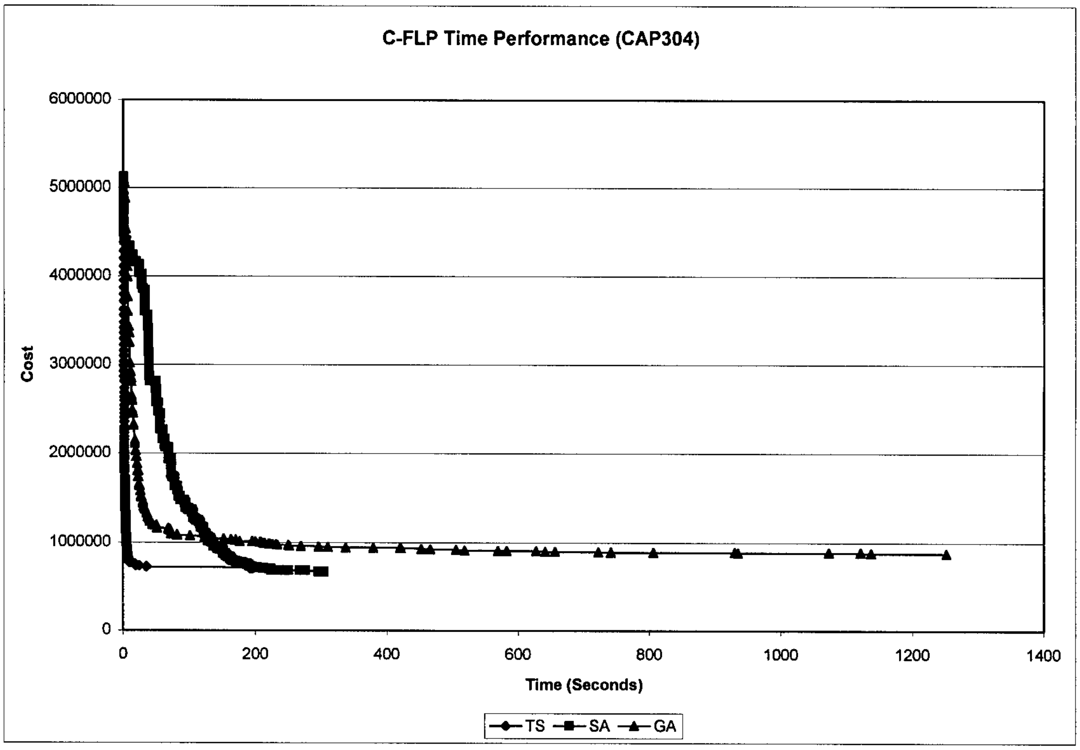

Figure 126. CFLP Time Performance (CAP304)

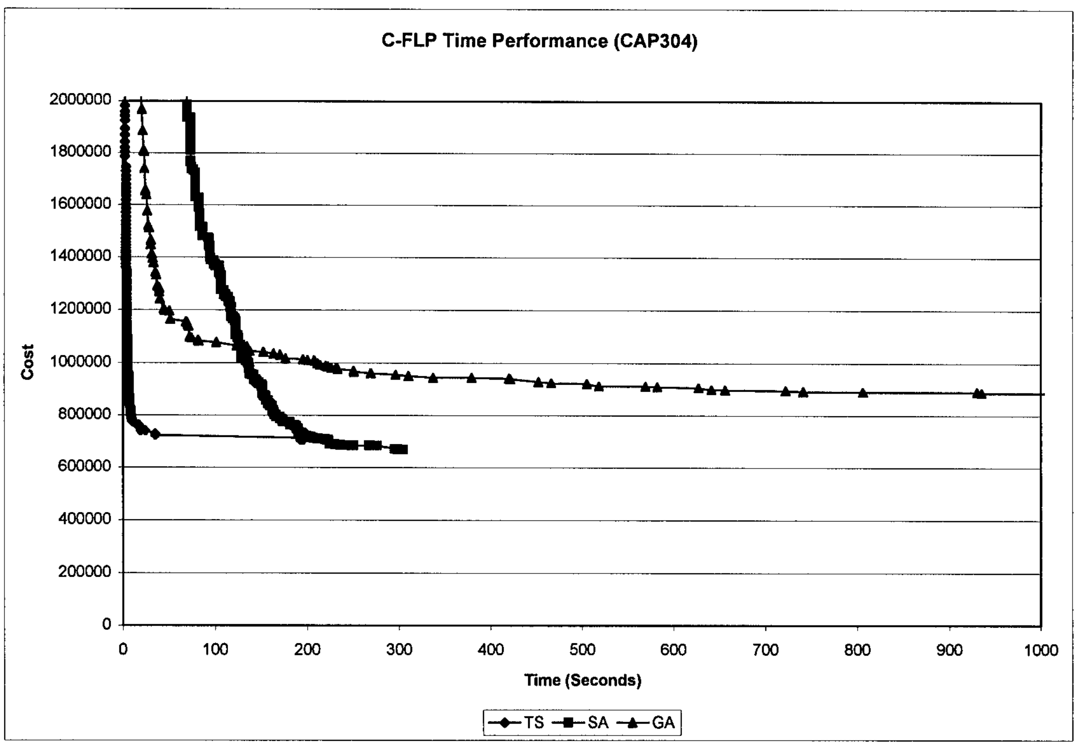

Figure 127. CFLP Time Performance Close-up (CAP304)

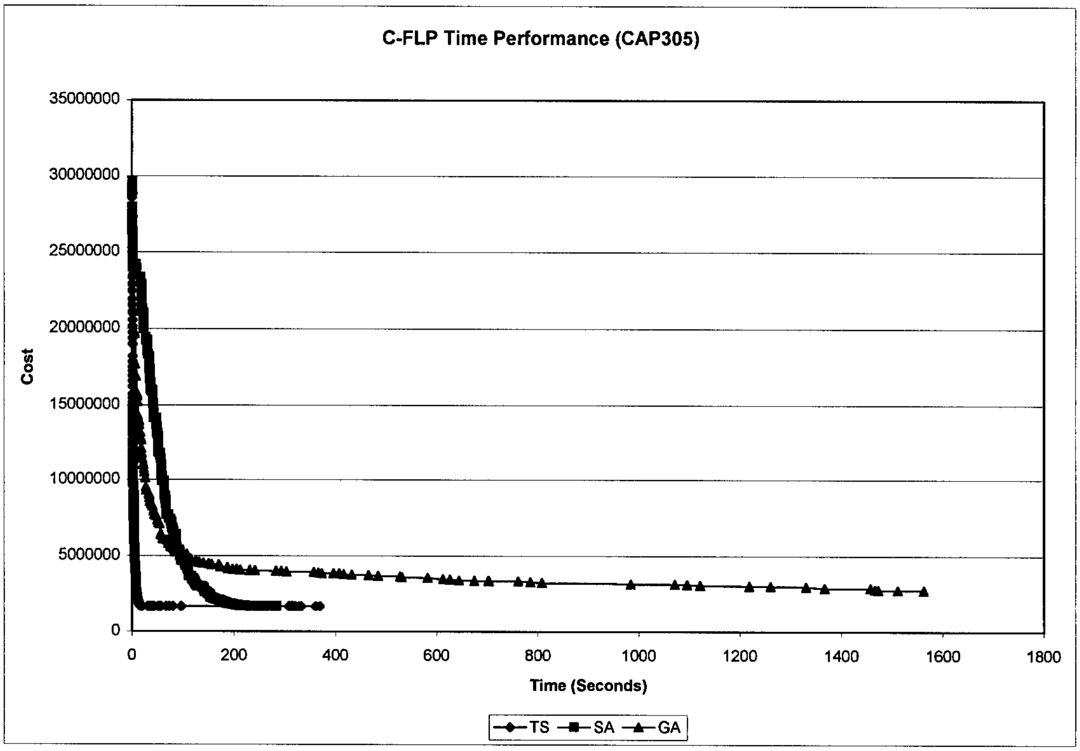

Figure 128. CFLP Time Performance (CAP305)

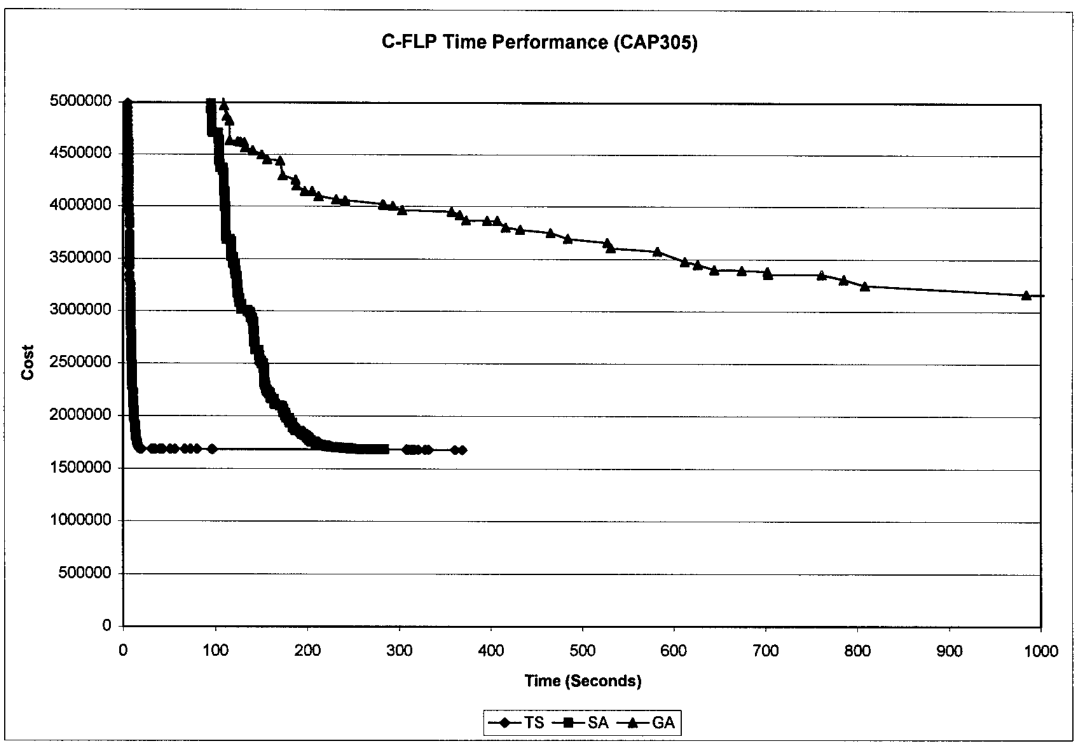

Figure 129. CFLP Time Performance Close-up (CAP305)

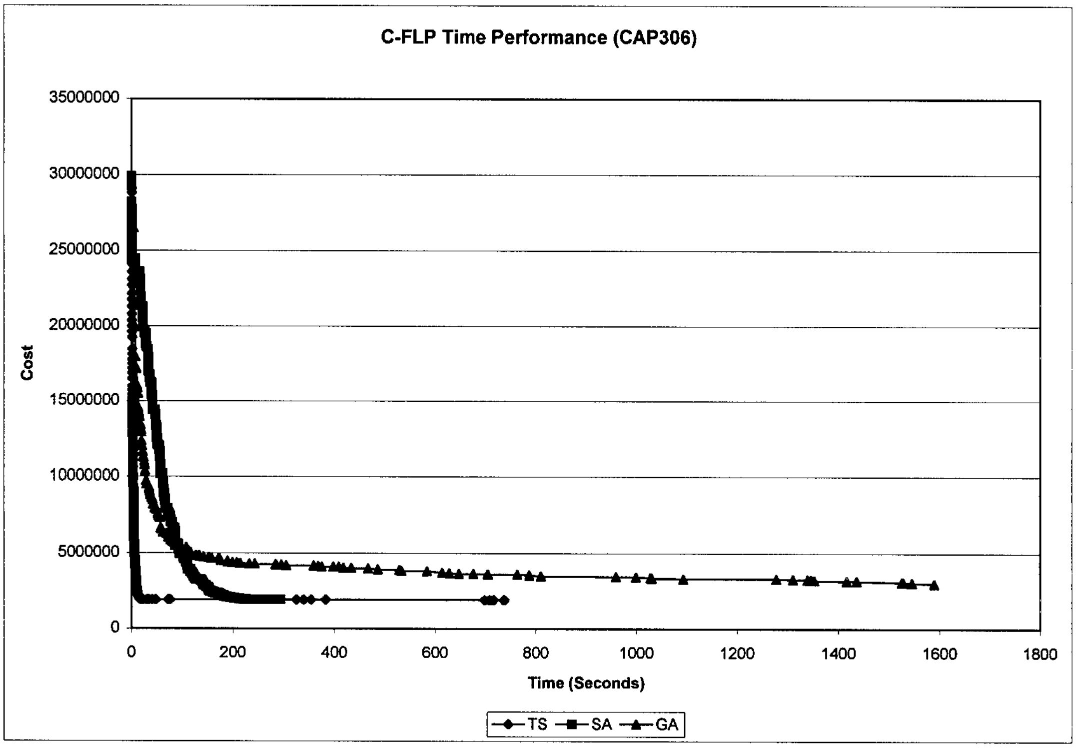

Figure 130. CFLP Time Performance (CAP306)

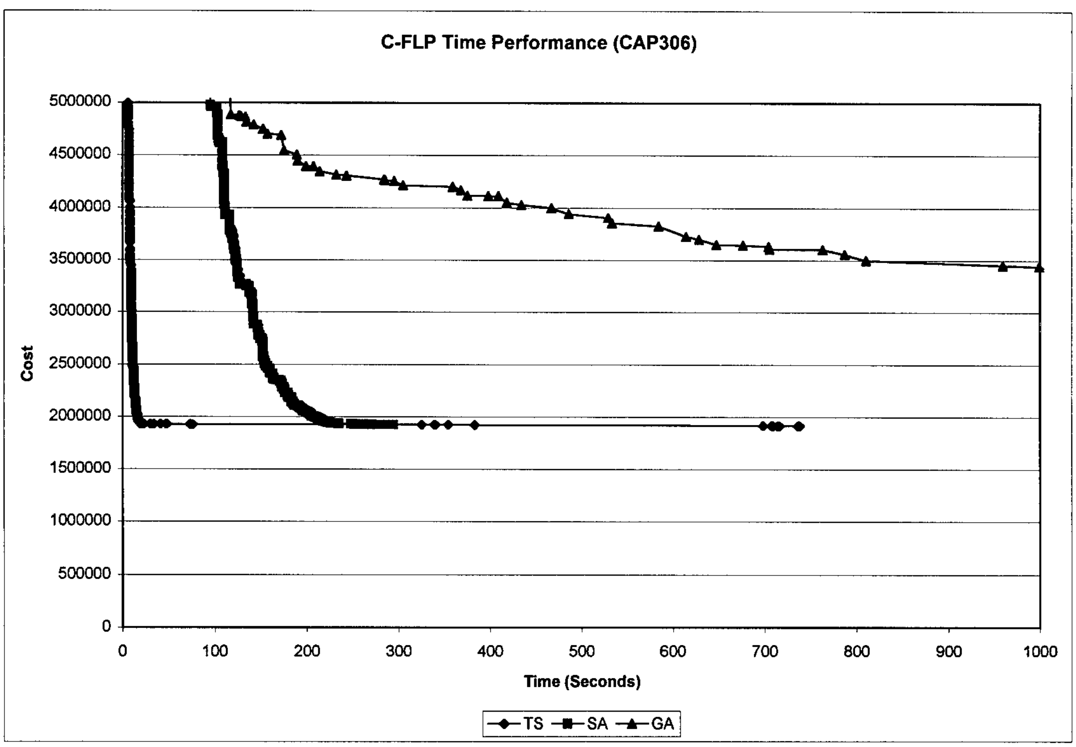

Figure 131. CFLP Time Performance Close-up (CAP306)

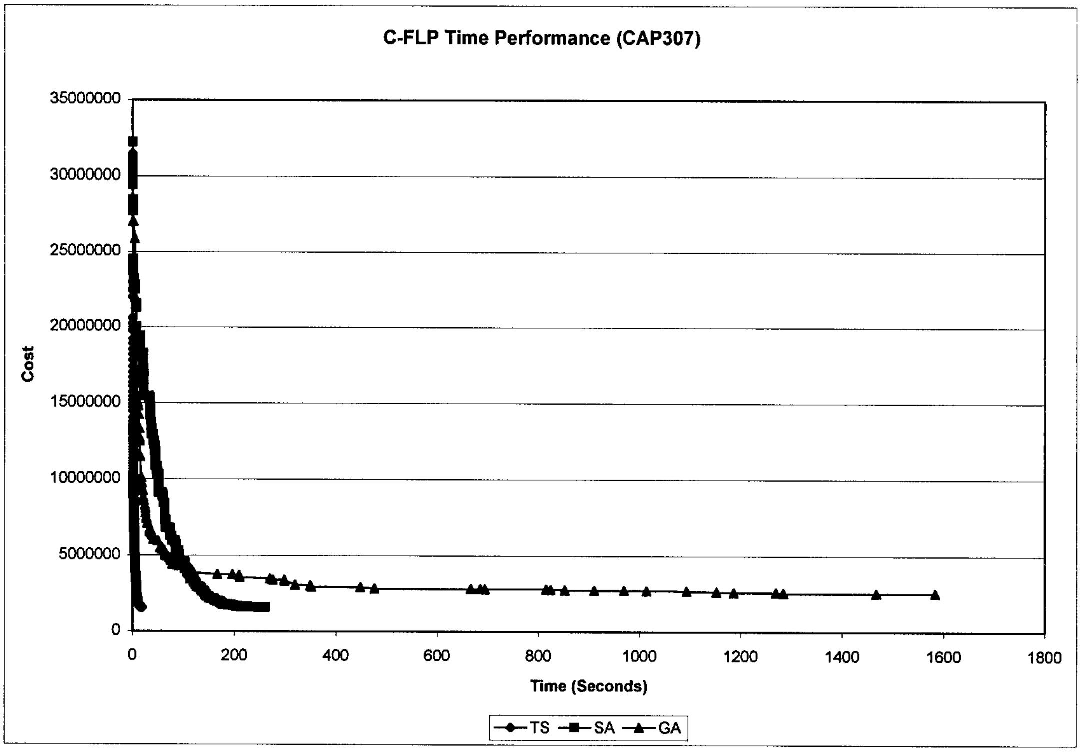

Figure 132. CFLP Time Performance (CAP307)

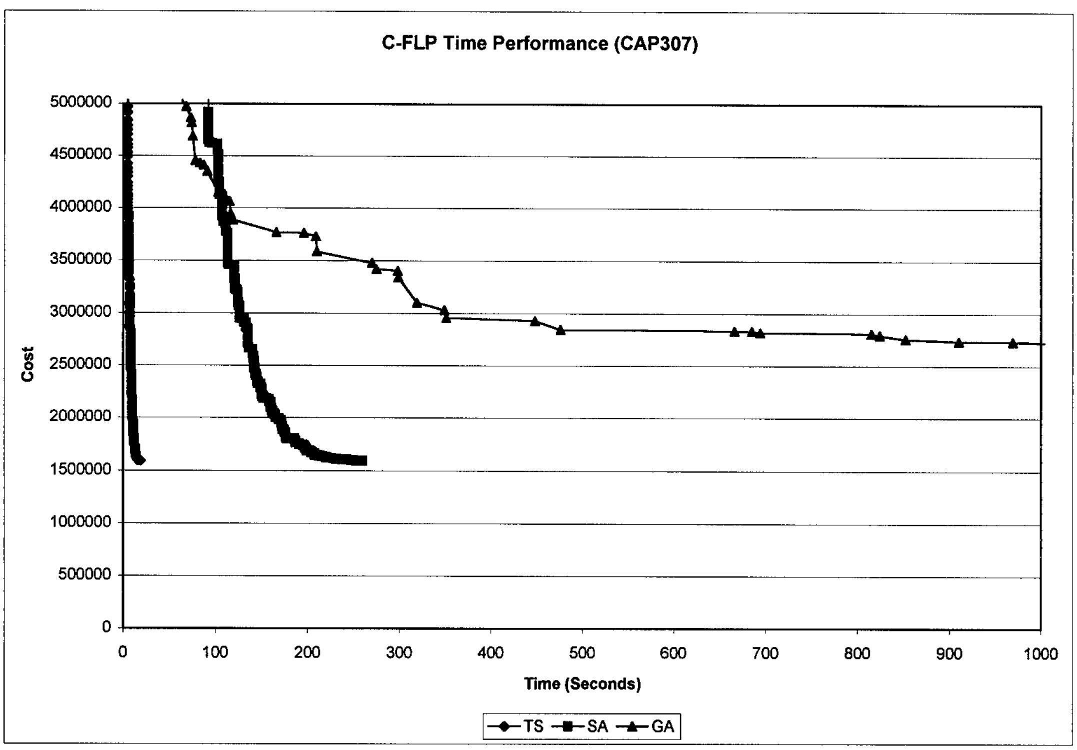

Figure 133. CFLP Time Performance Close-up (CAP307)

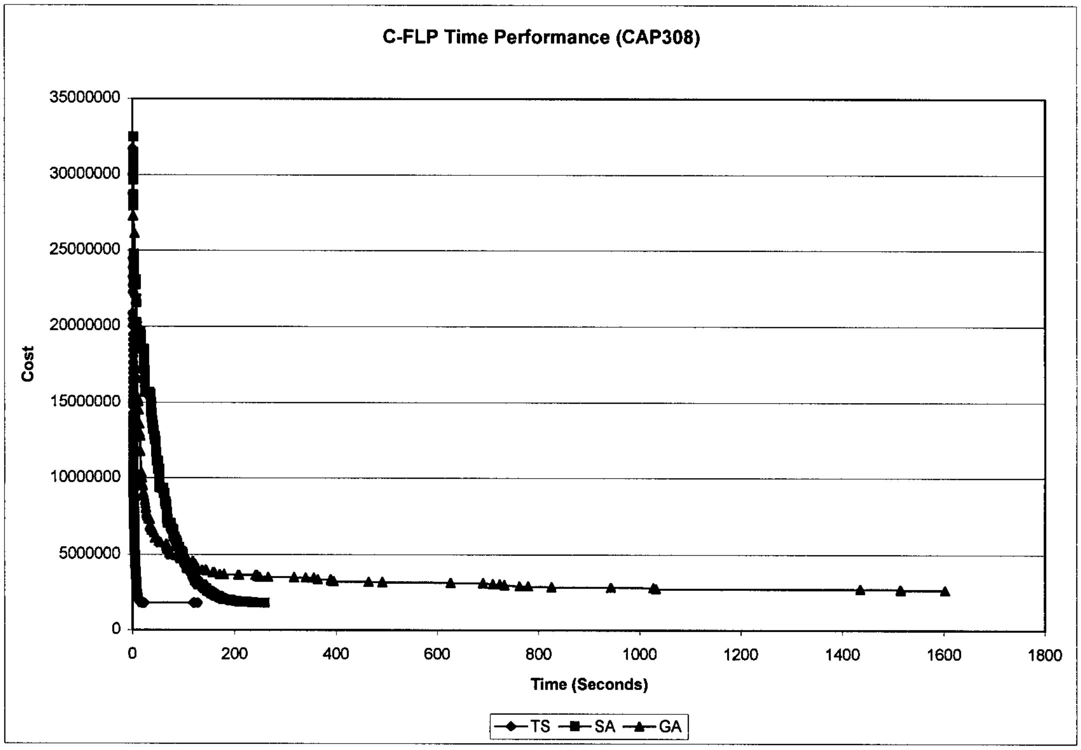

Figure 134. CFLP Time Performance (CAP308)

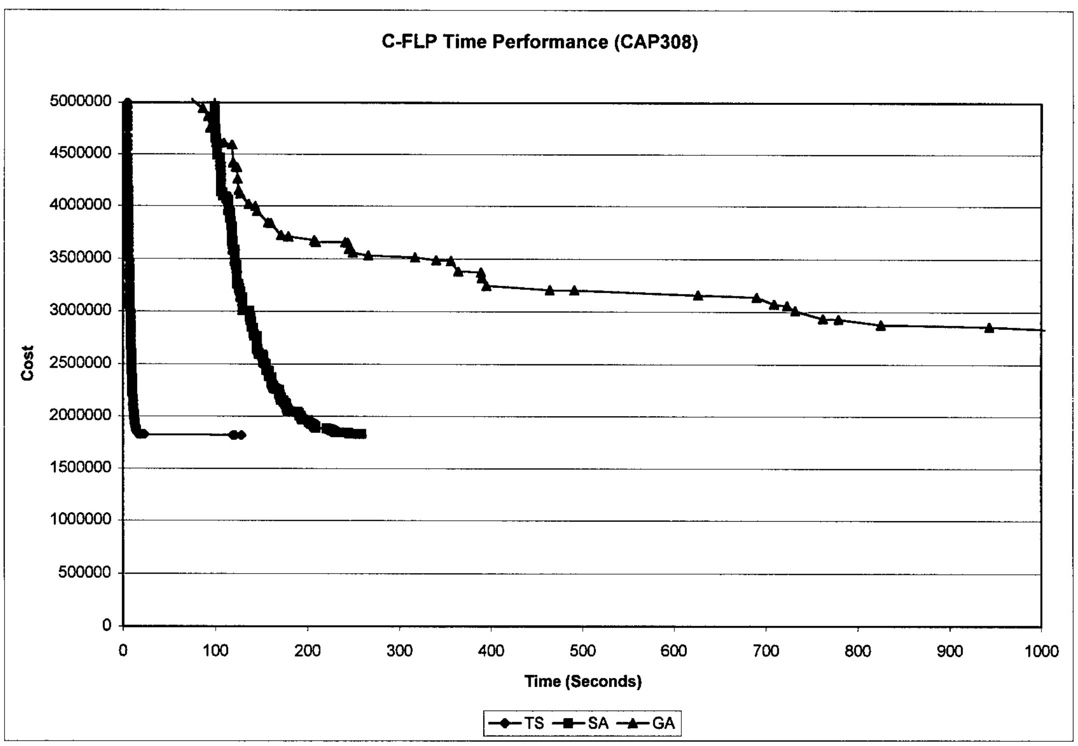

Figure 135. CFLP Time Performance Close-up (CAP308)

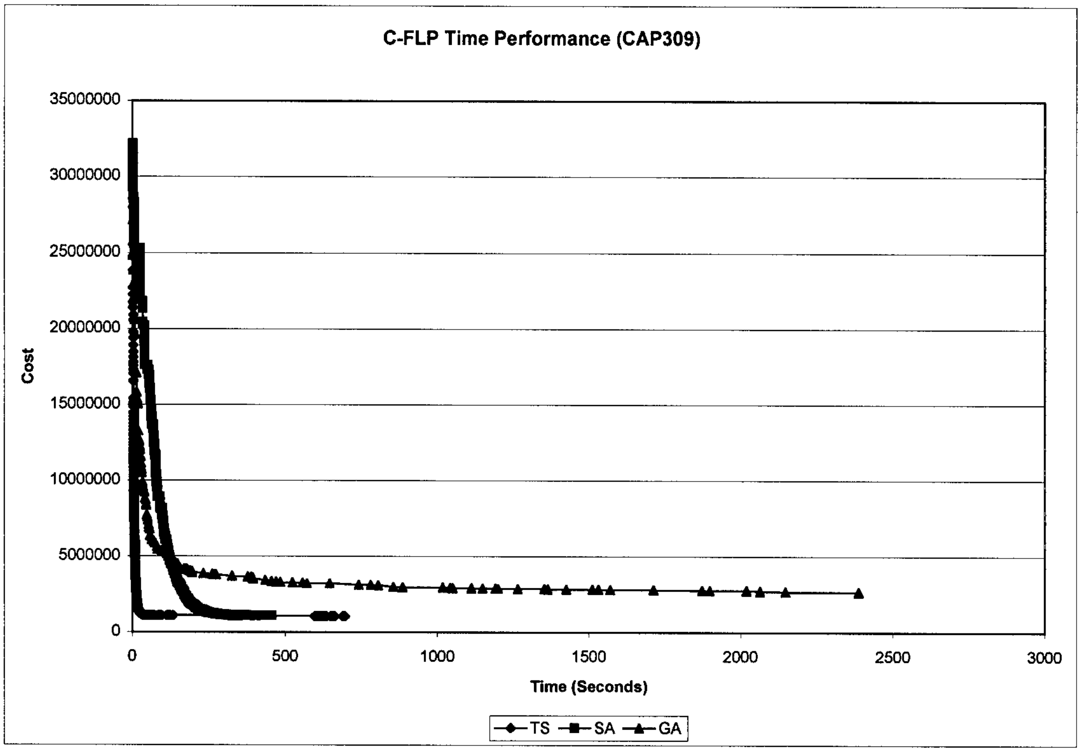

Figure 136. CFLP Time Performance (CAP309)

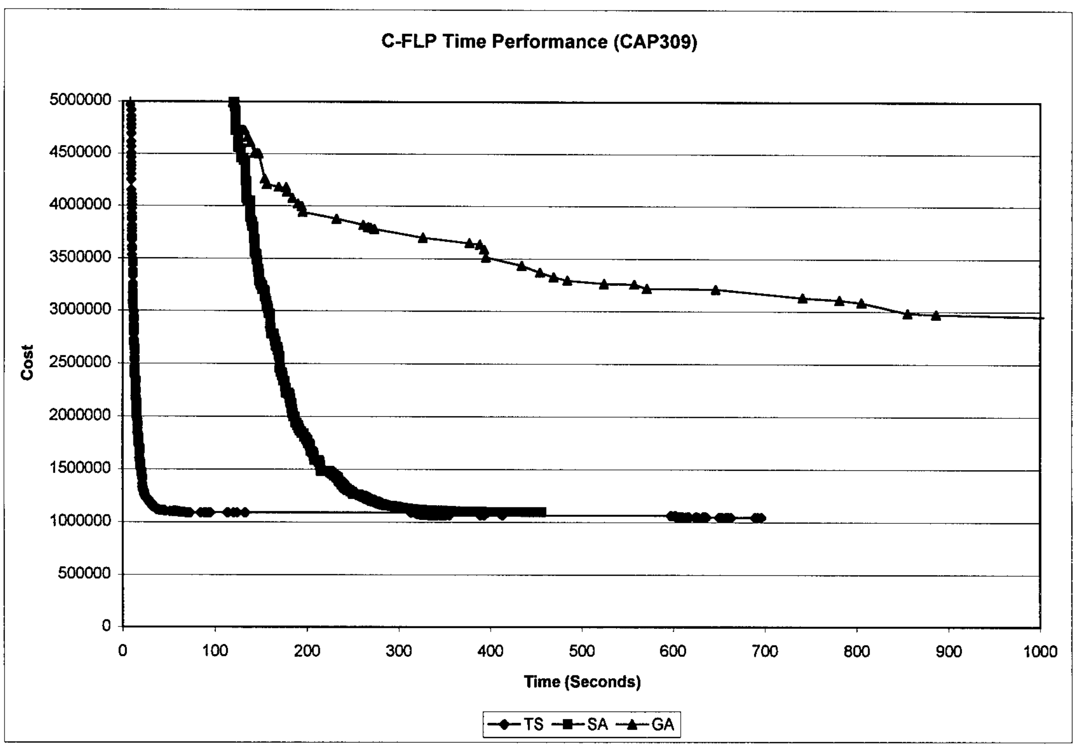

Figure 137. CFLP Time Performance Close-up (CAP309)

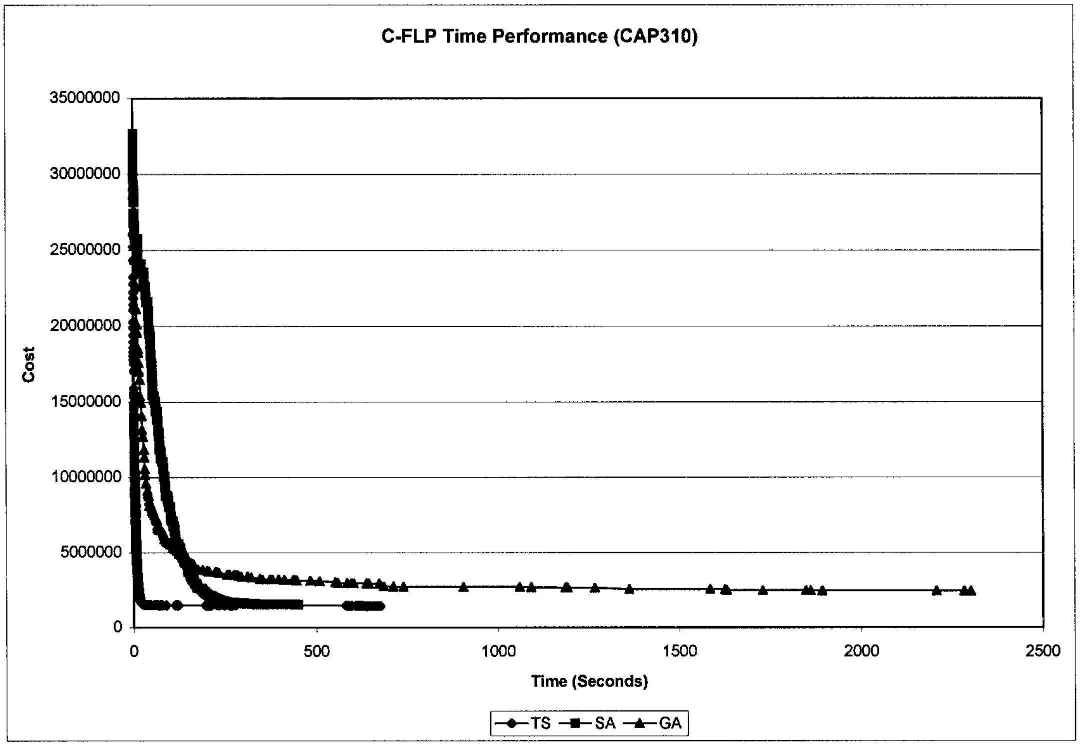

Figure 138. CFLP Time Performance (CAP310)

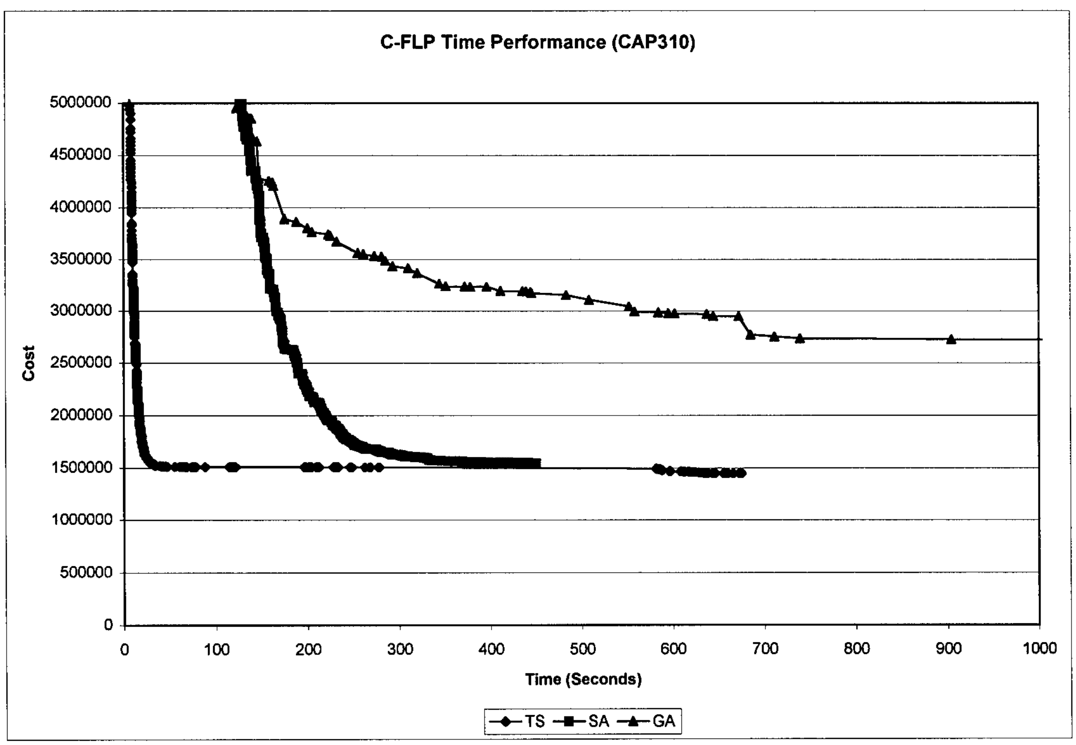

Figure 139. CFLP Time Performance Close-up (CAP310)

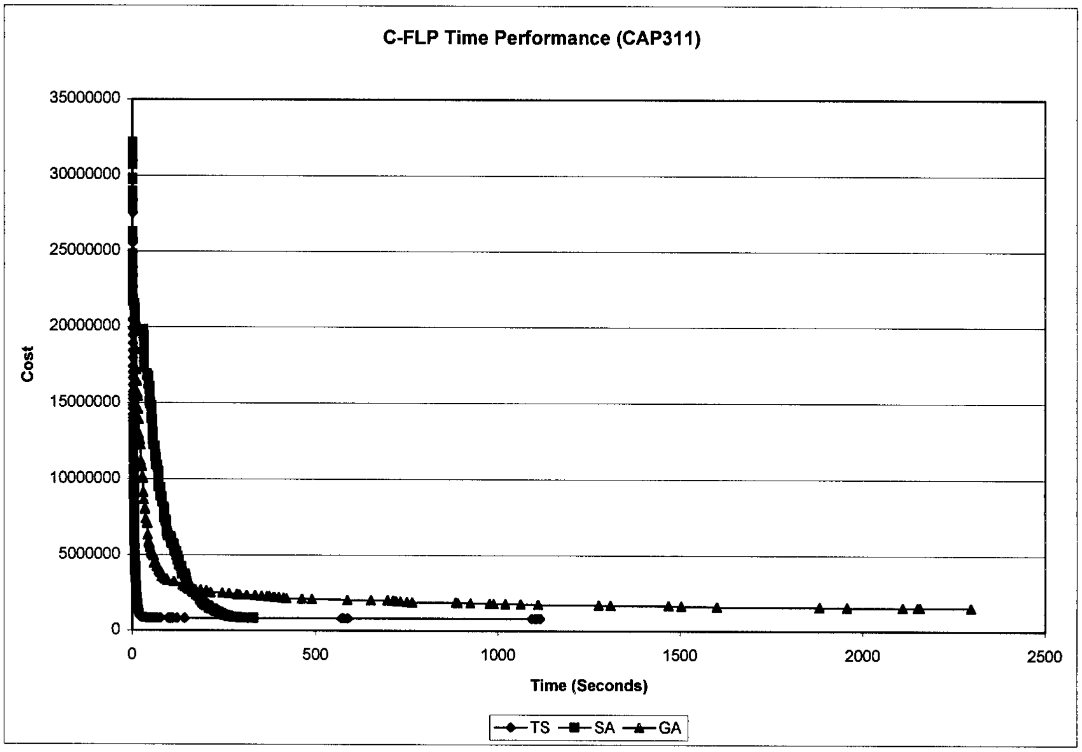

Figure 140. CFLP Time Performance (CAP311)

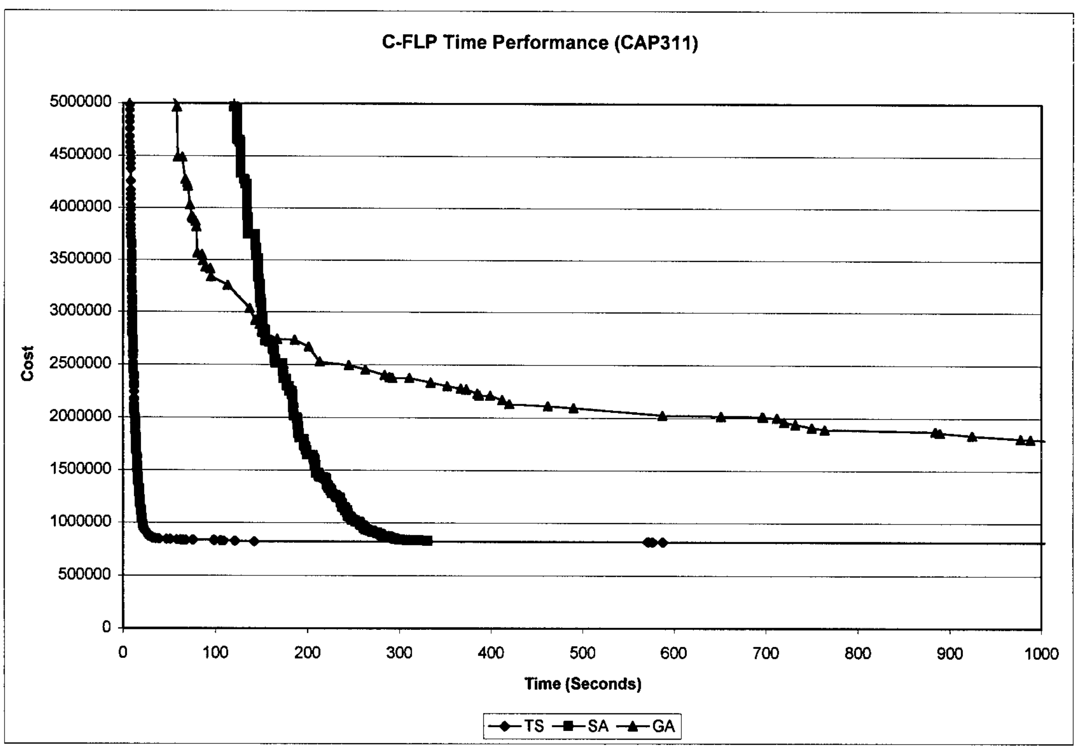

Figure 141. CFLP Time Performance Close-up (CAP311)

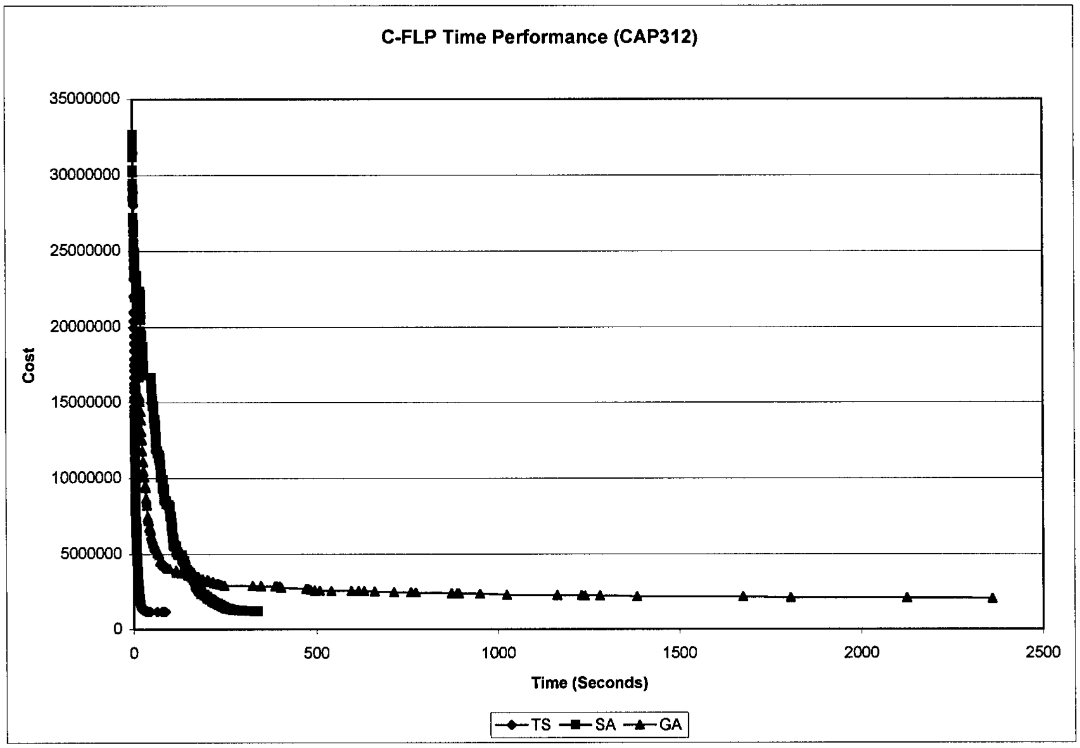

Figure 142. CFLP Time Performance (CAP312)

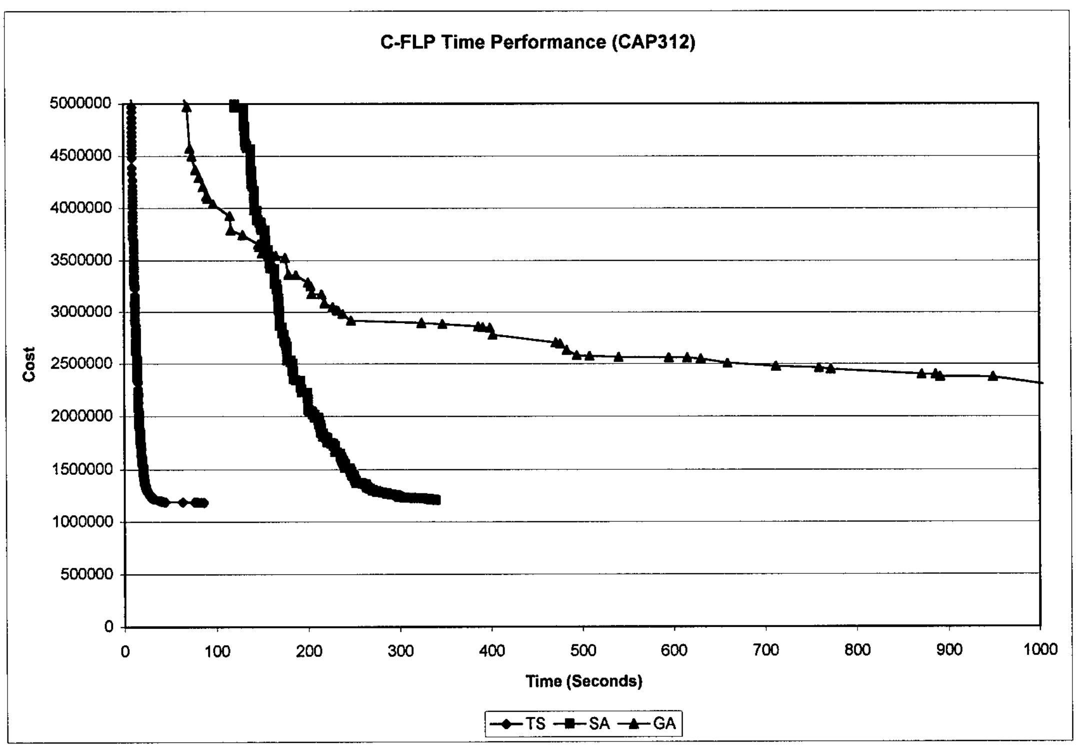

Figure 143. CFLP Time Performance Close-up (CAP312)

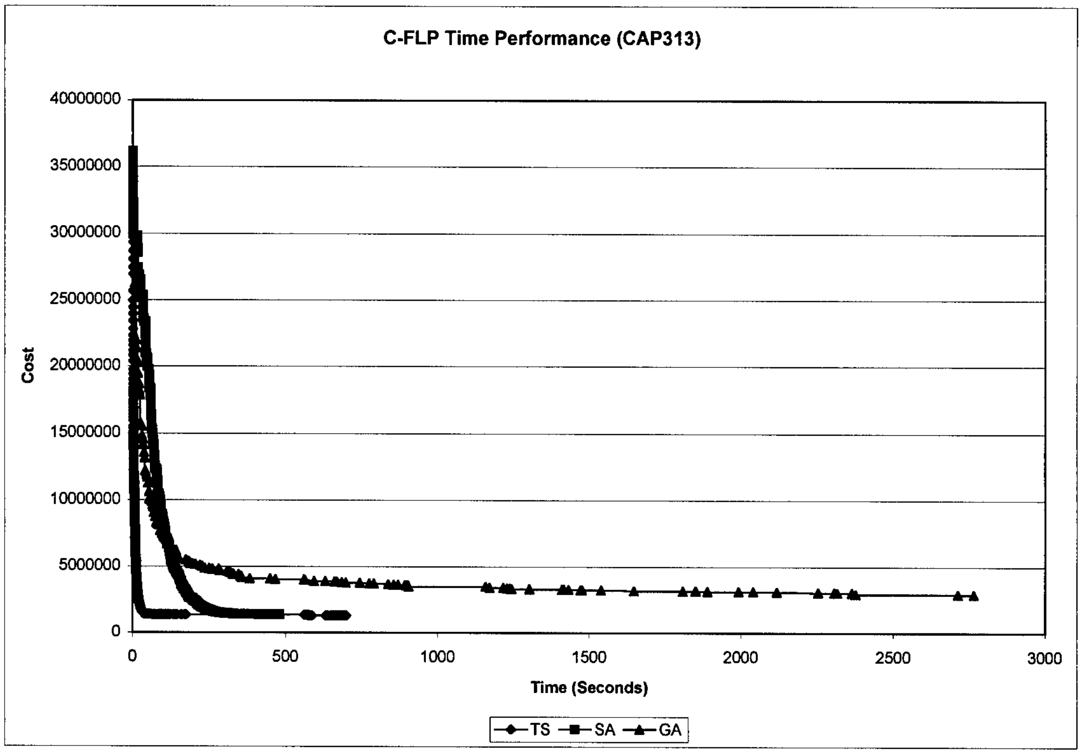

Figure 144. CFLP Time Performance (CAP313)

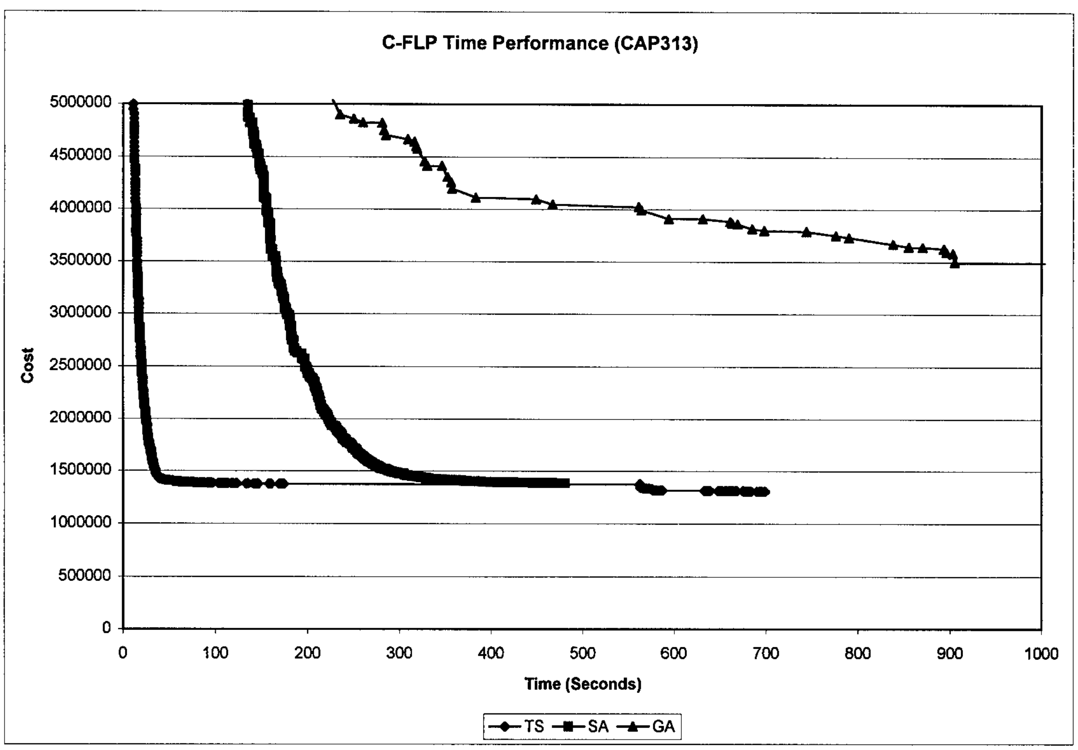

Figure 145. CFLP Time Performance Close-up (CAP313)

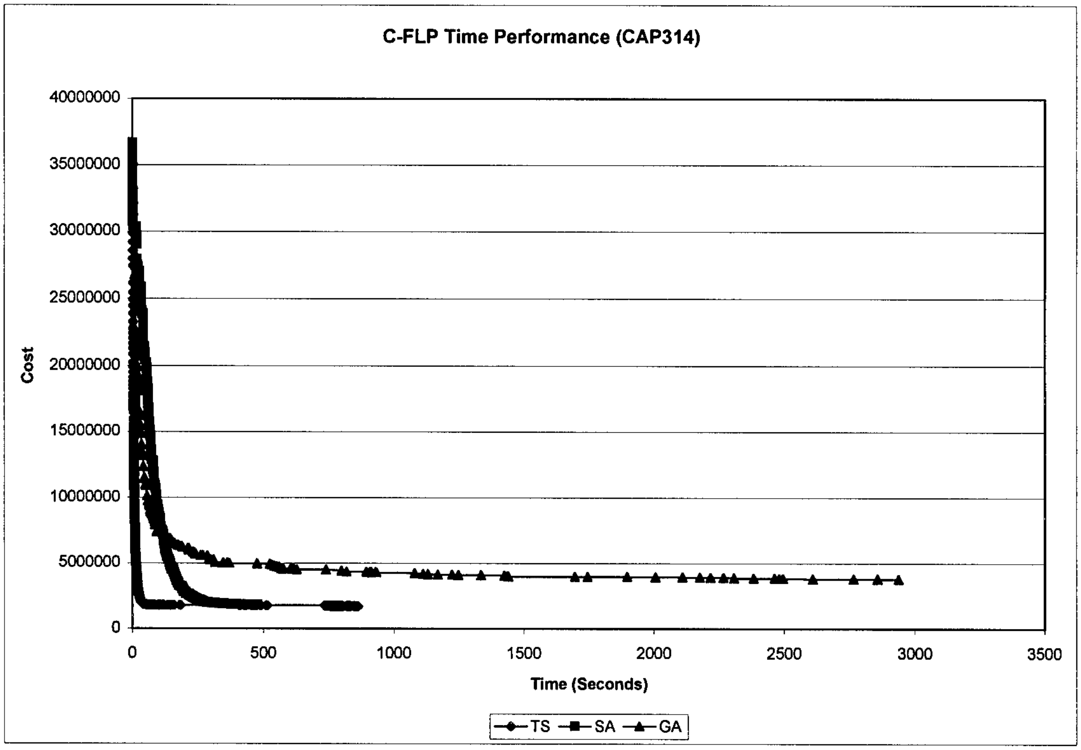

Figure 146. CFLP Time Performance (CAP314)

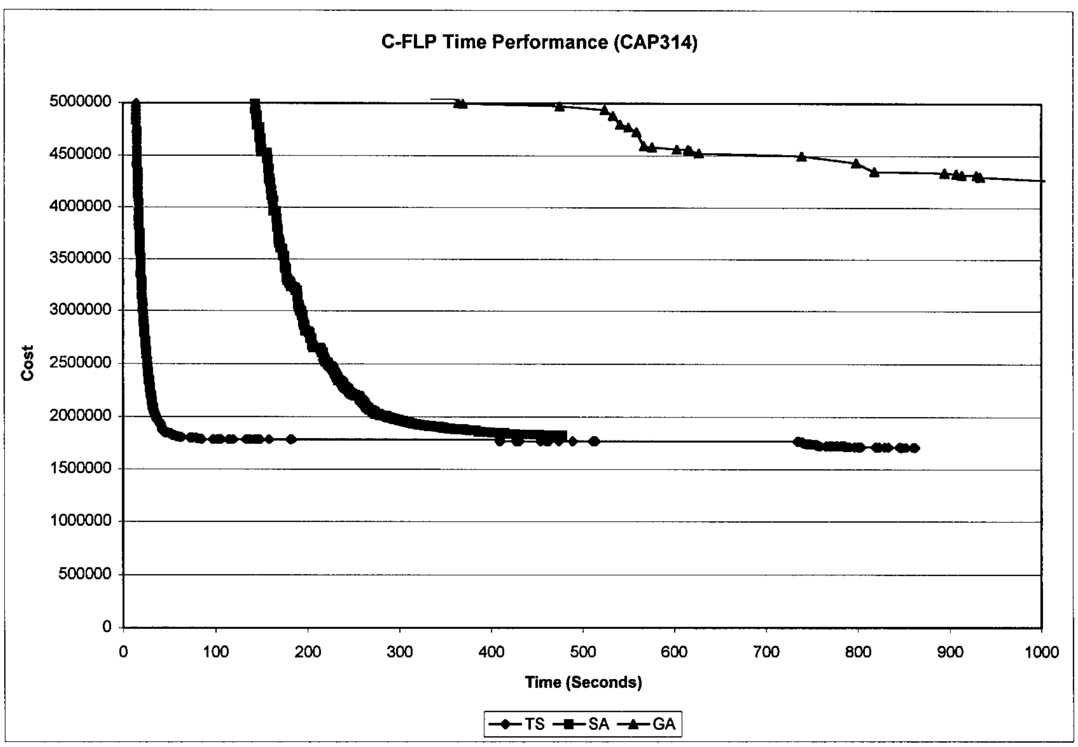

Figure 147. CFLP Time Performance Close-up (CAP314)

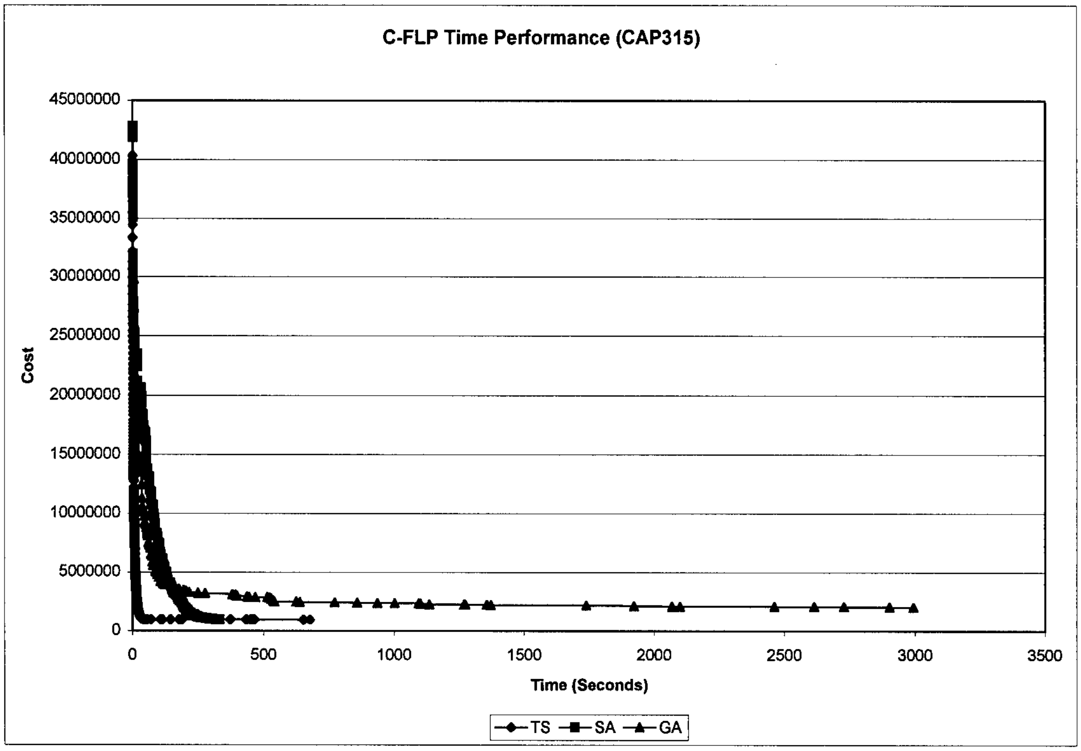

Figure 148. CFLP Time Performance (CAP315)

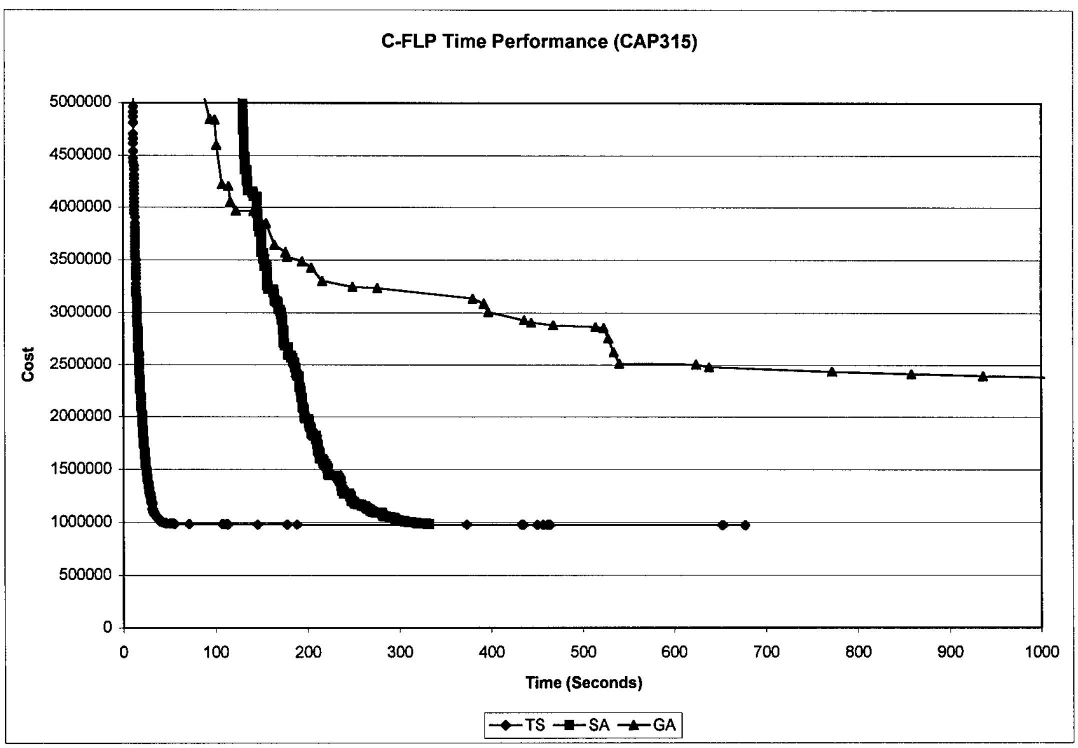

Figure 149. CFLP Time Performance Close-up (CAP315)

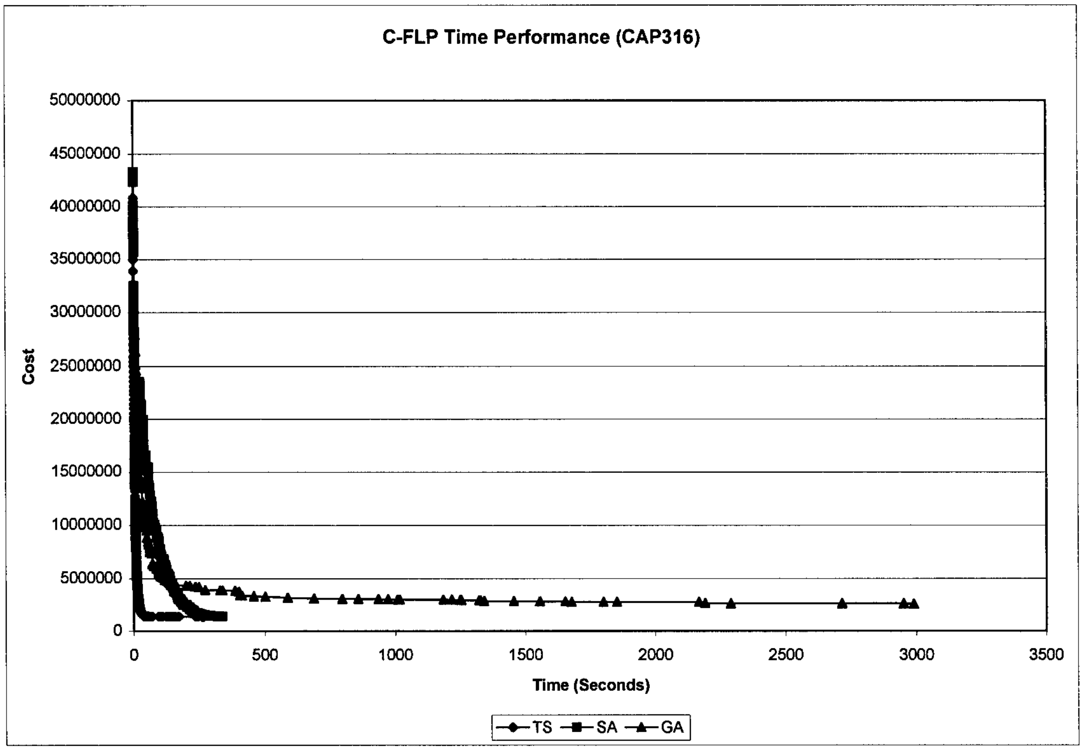

Figure 150. CFLP Time Performance (CAP316)

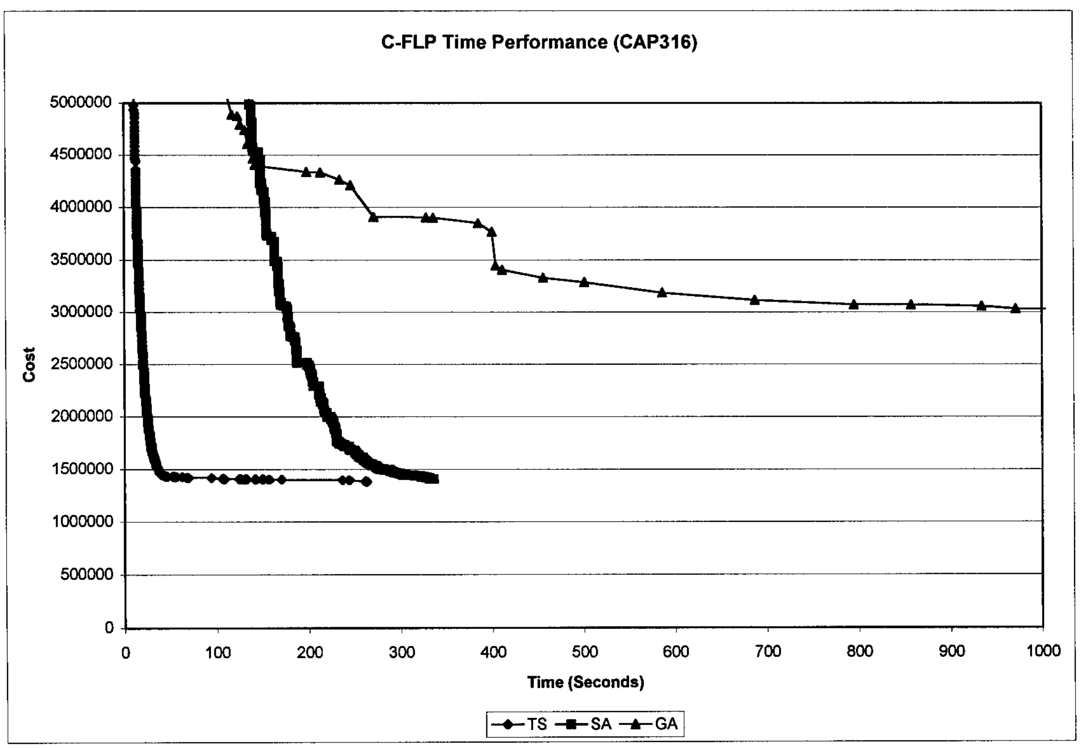

Figure 151. CFLP Time Performance Close-up (CAP316)

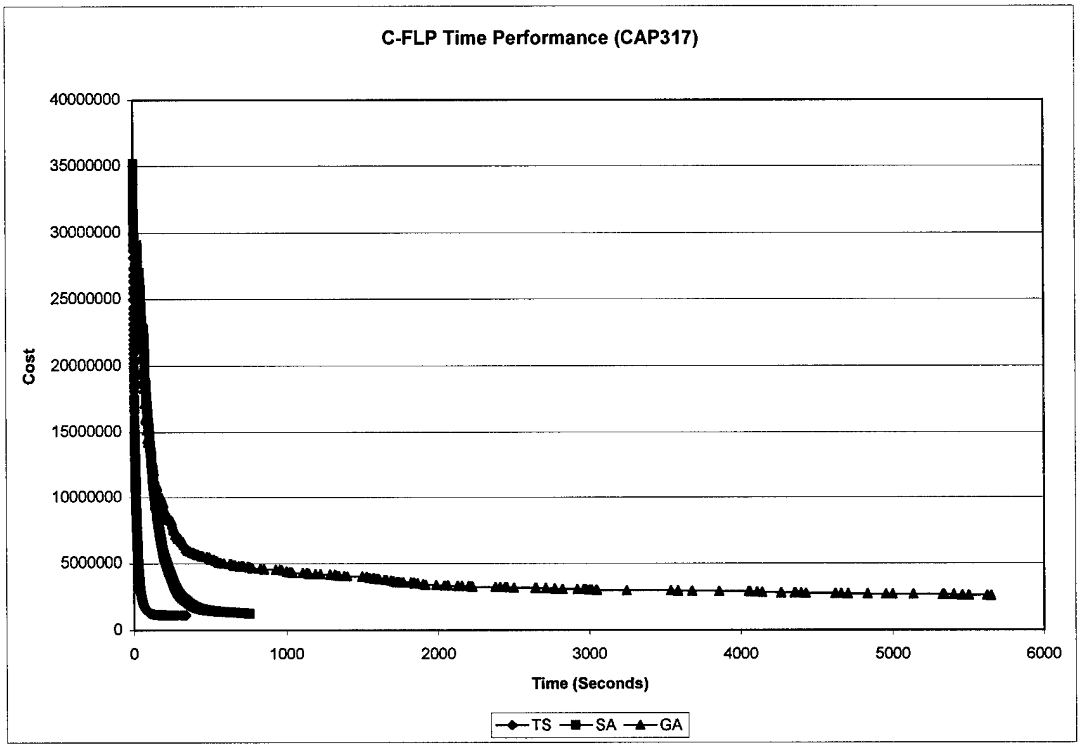

Figure 152. CFLP Time Performance (CAP317)

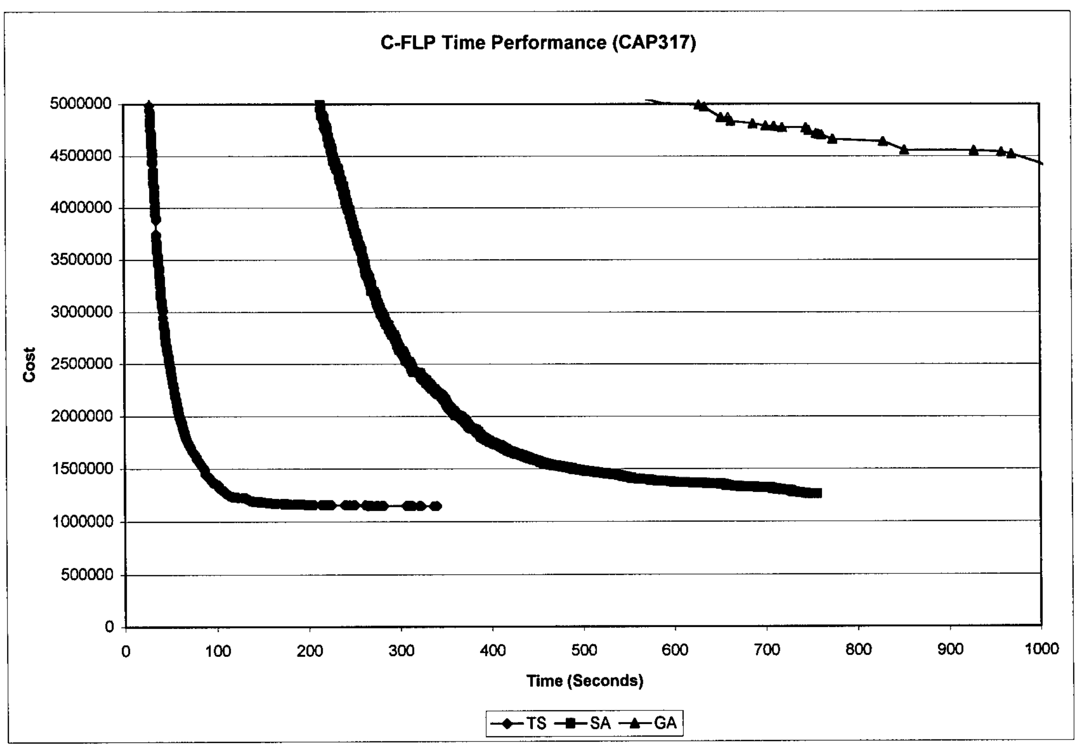

Figure 153. CFLP Time Performance Close-up (CAP317)

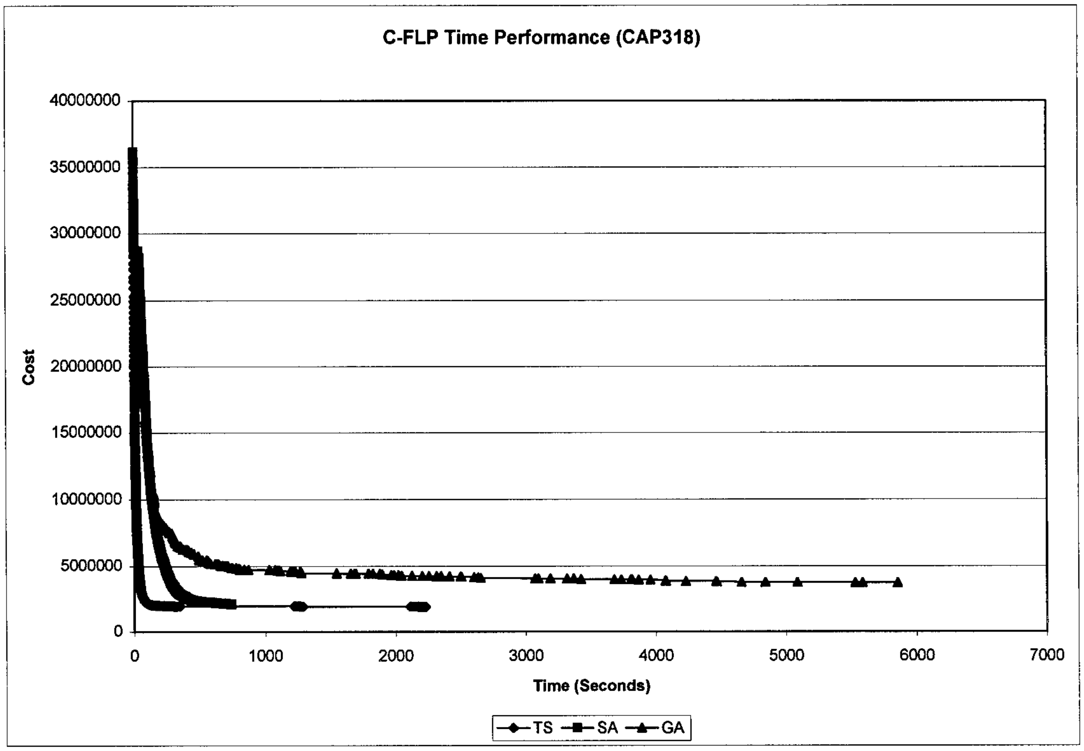

Figure 154. CFLP Time Performance (CAP318)

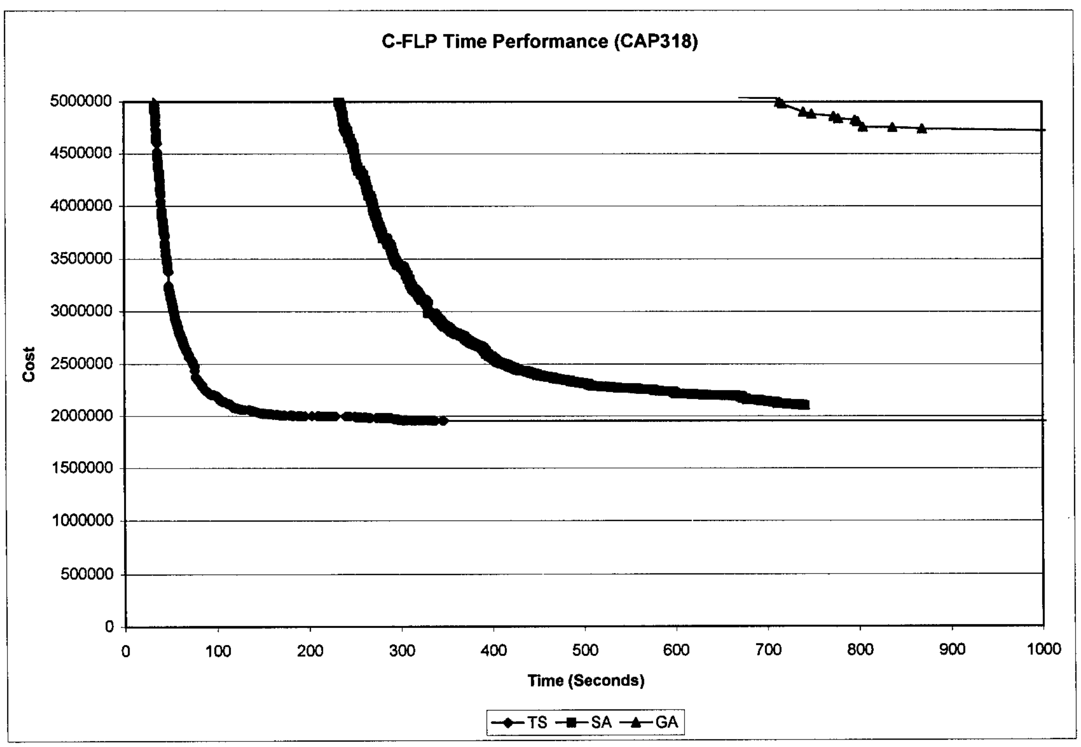

Figure 155. CFLP Time Performance Close-up (CAP318)

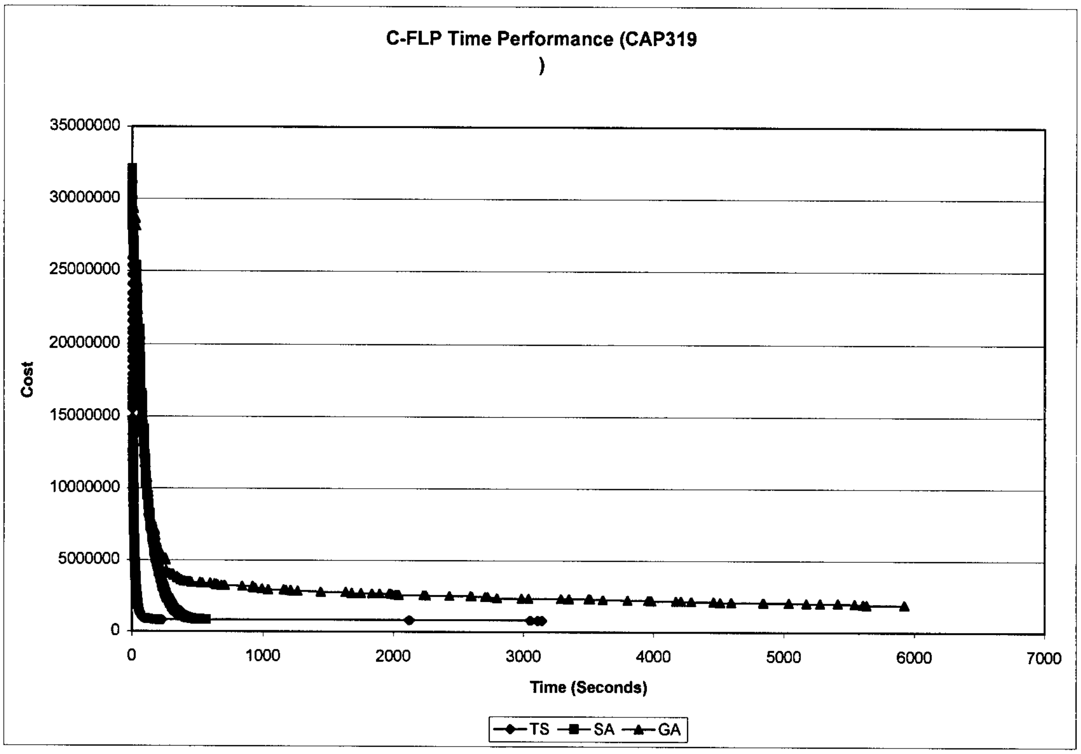

Figure 156. CFLP Time Performance (CAP319)

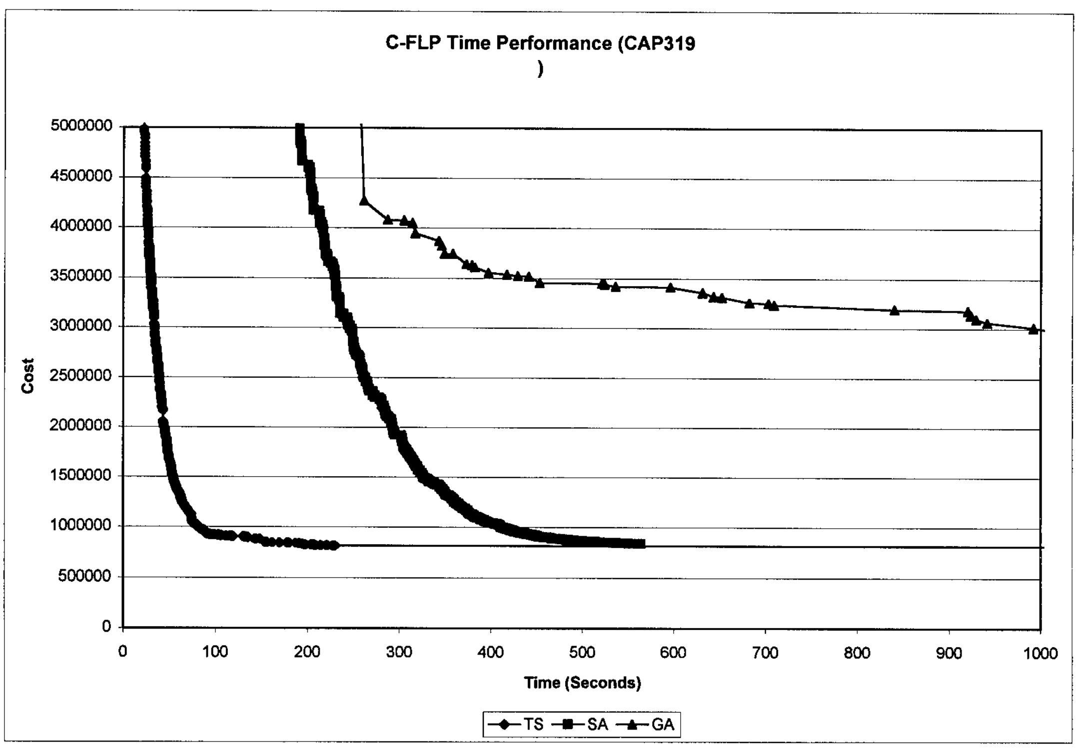

Figure 157. CFLP Time Performance Close-up (CAP319)

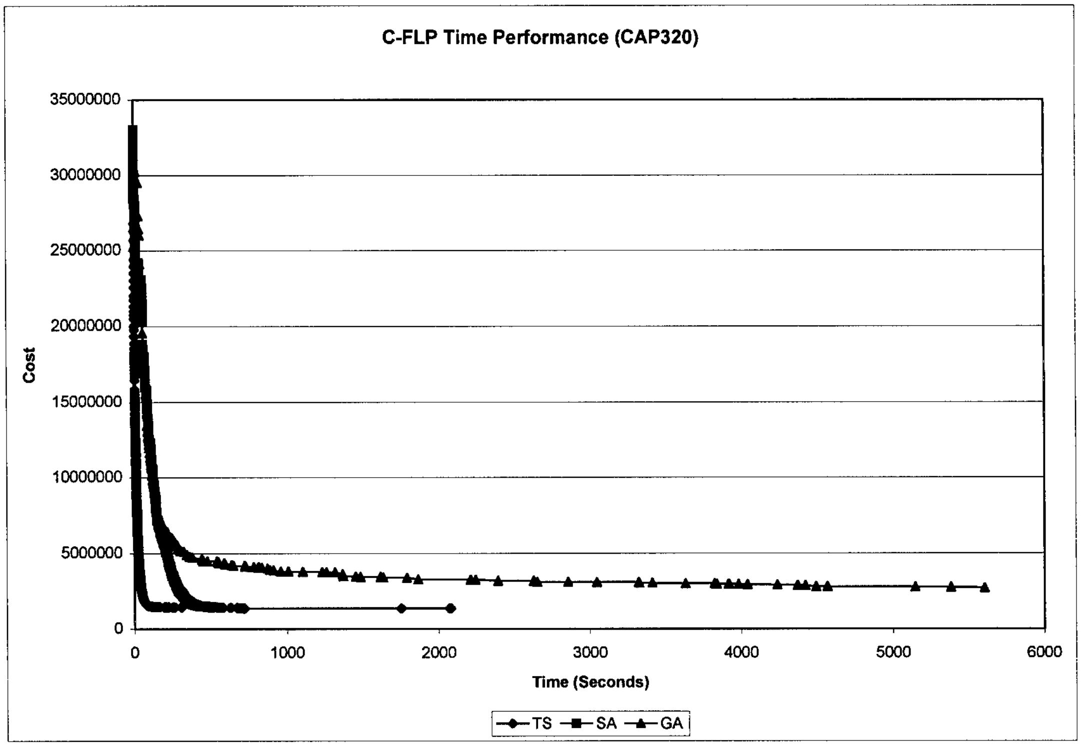

Figure 158. CFLP Time Performance (CAP320)

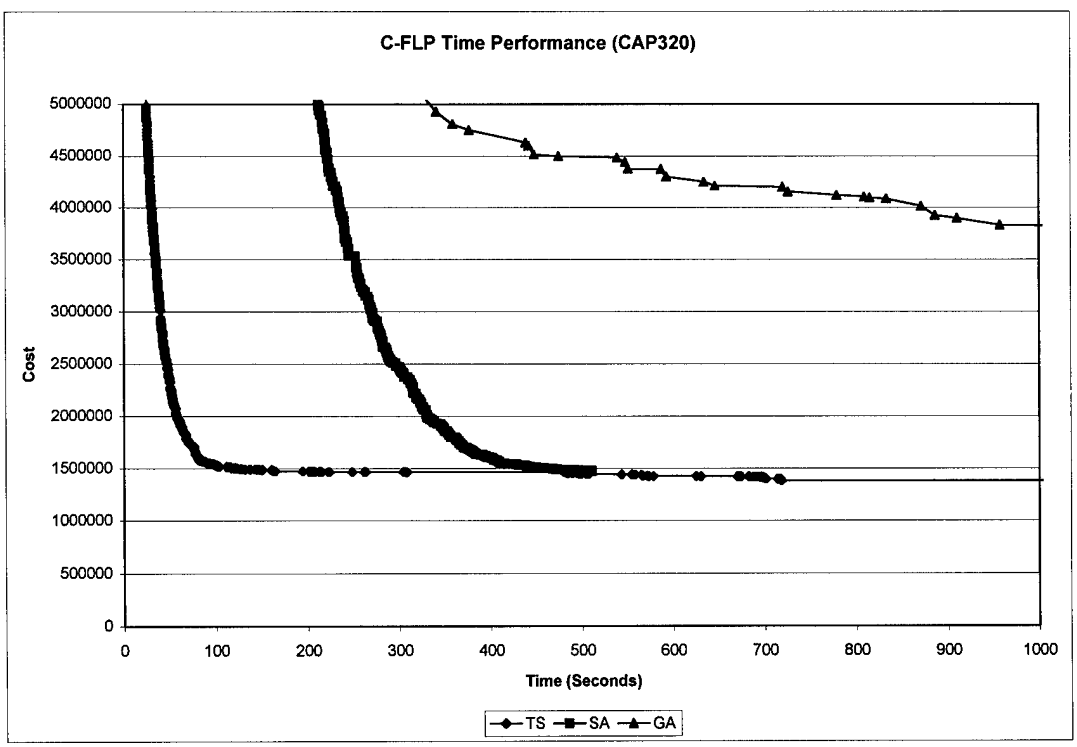

Figure 159. CFLP Time Performance Close-up (CAP320)

APPENDIX F

SOLUTIONS PERFORMANCE CHARTS FOR CFLP

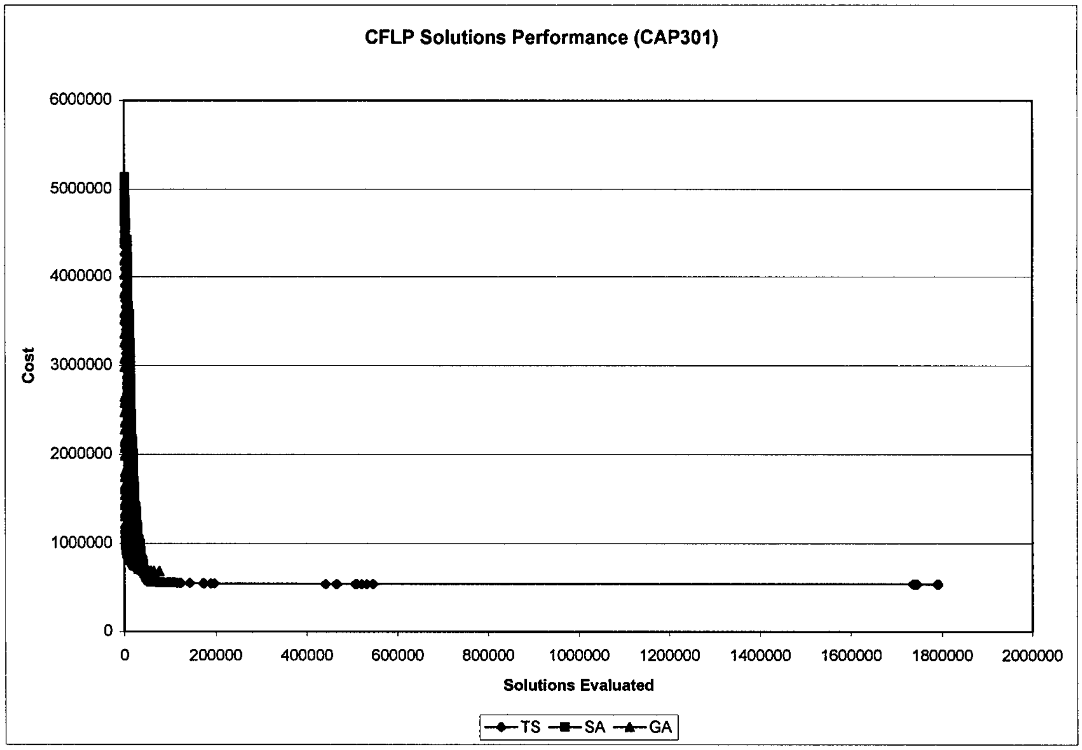

Figure 160. CFLP Solutions Performance (CAP301)

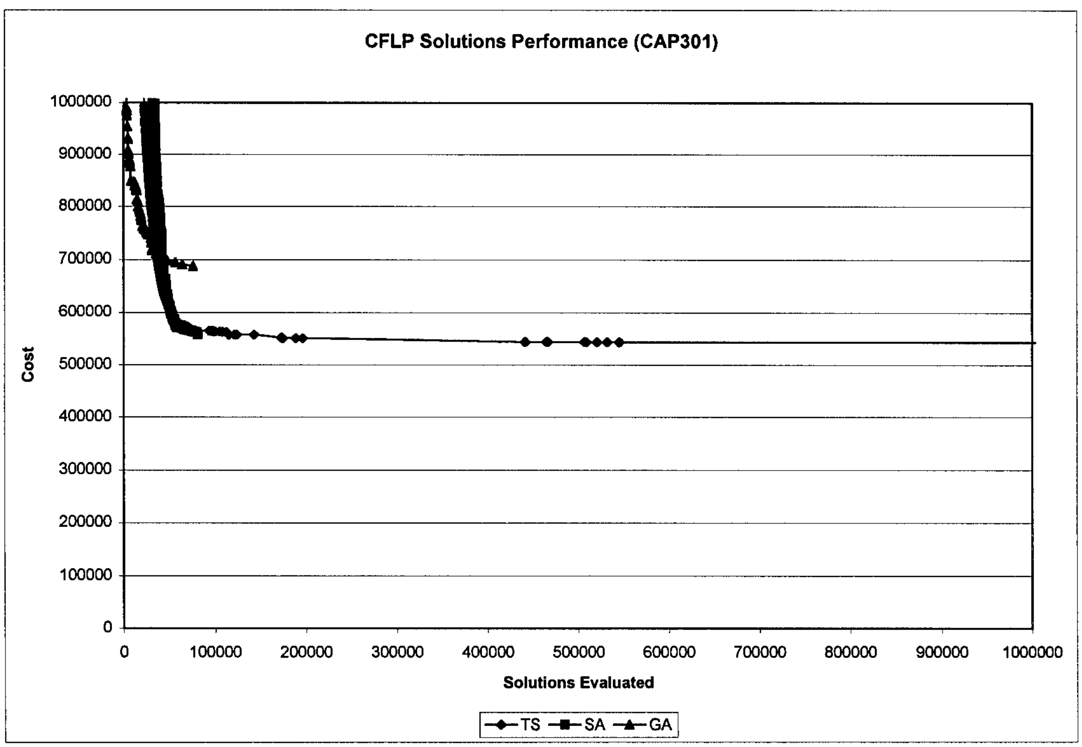

Figure 161. CFLP Solutions Performance Close-up (CAP301)

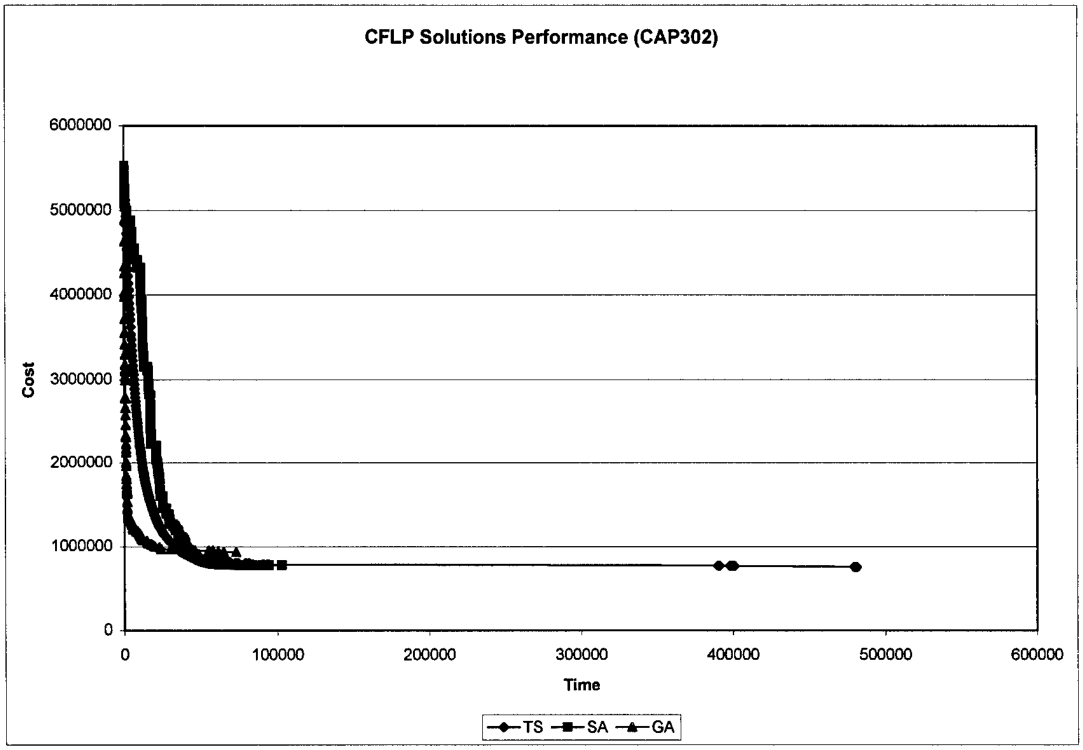

Figure 162. CFLP Solutions Performance (CAP302)

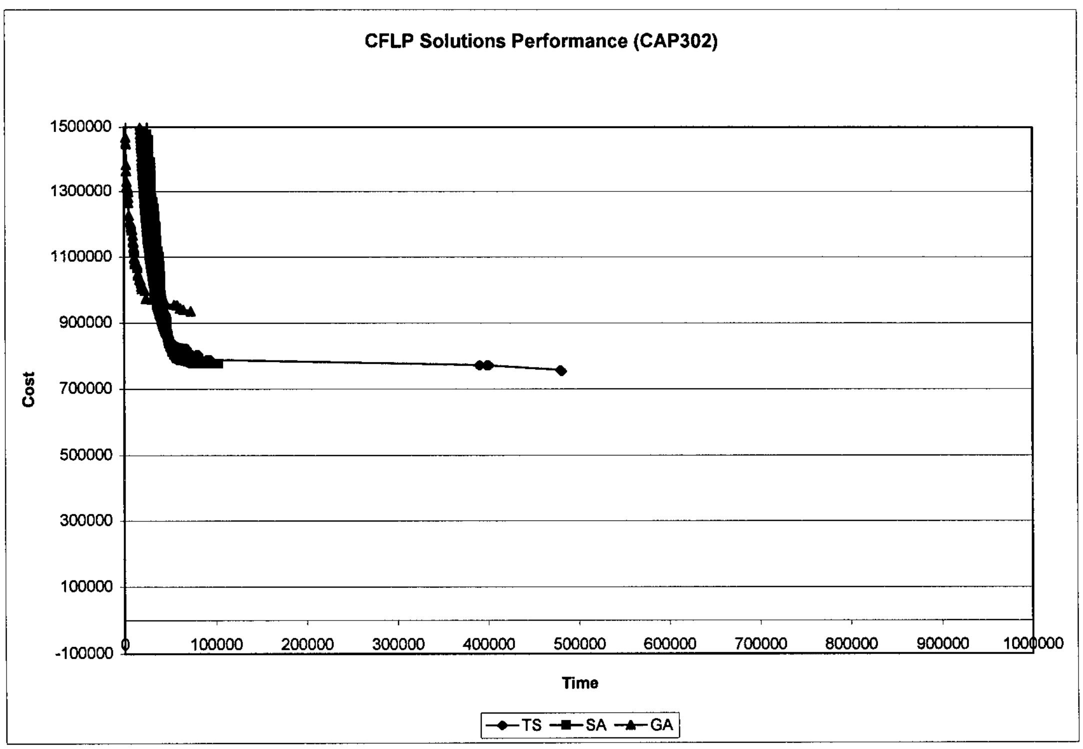

Figure 163. CFLP Solutions Performance Close-up (CAP302)

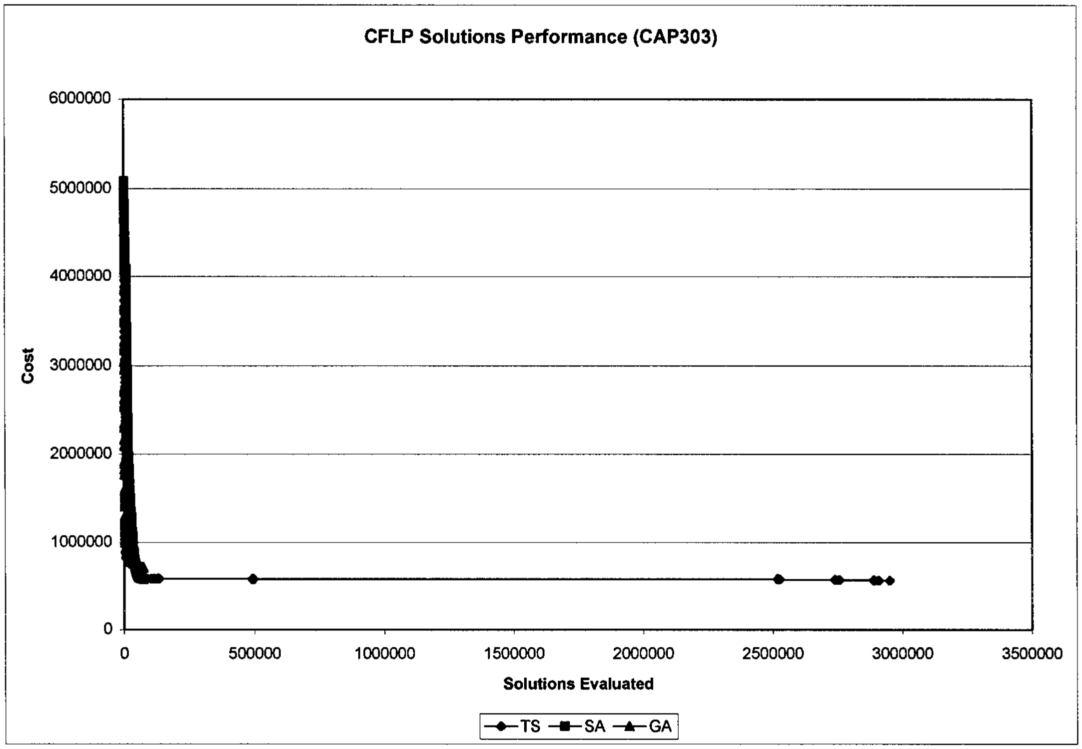

Figure 164. CFLP Solutions Performance (CAP303)

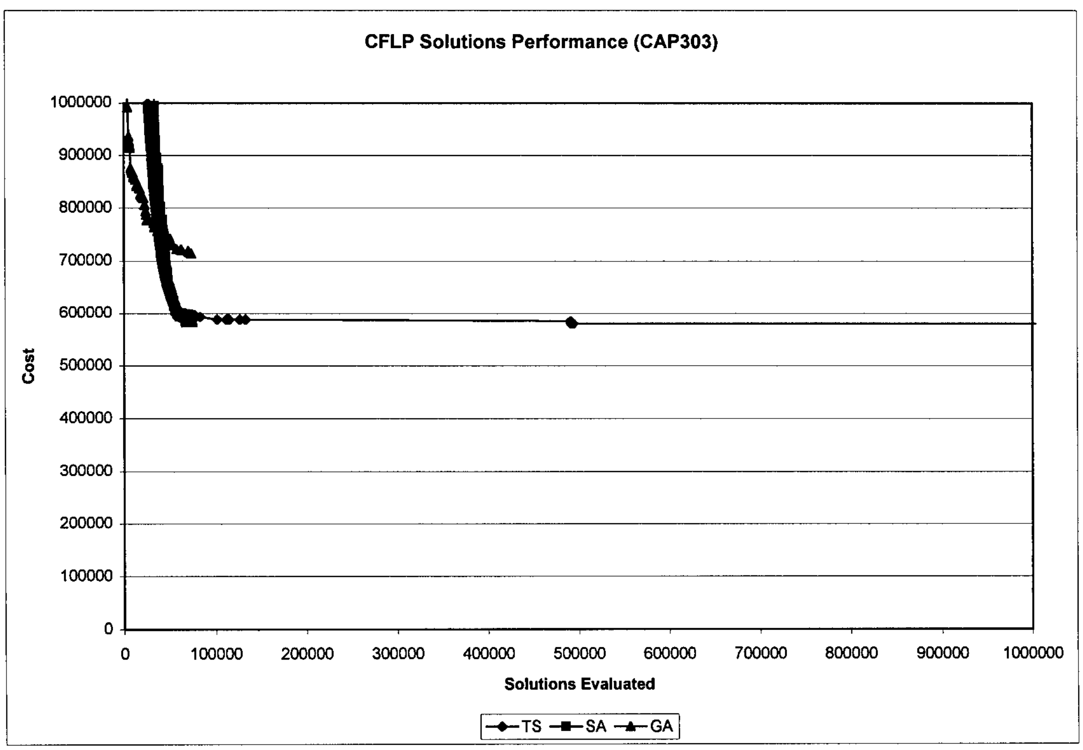

Figure 165. CFLP Solutions Performance Close-up (CAP303)

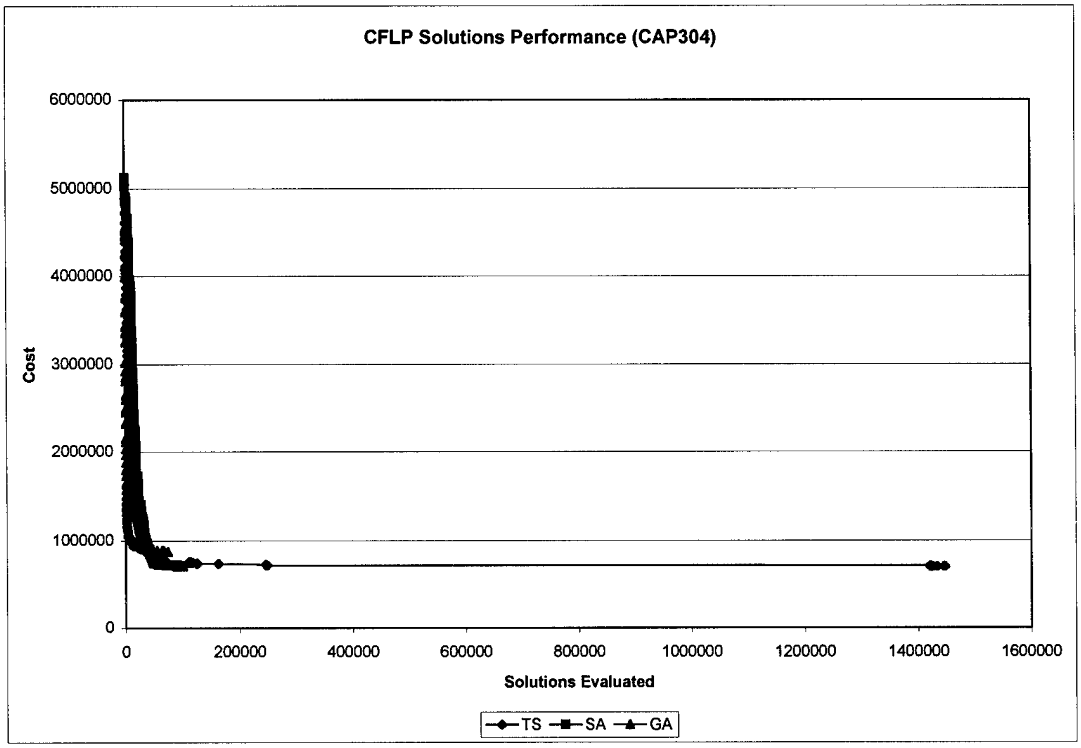

Figure 166. CFLP Solutions Performance (CAP304)

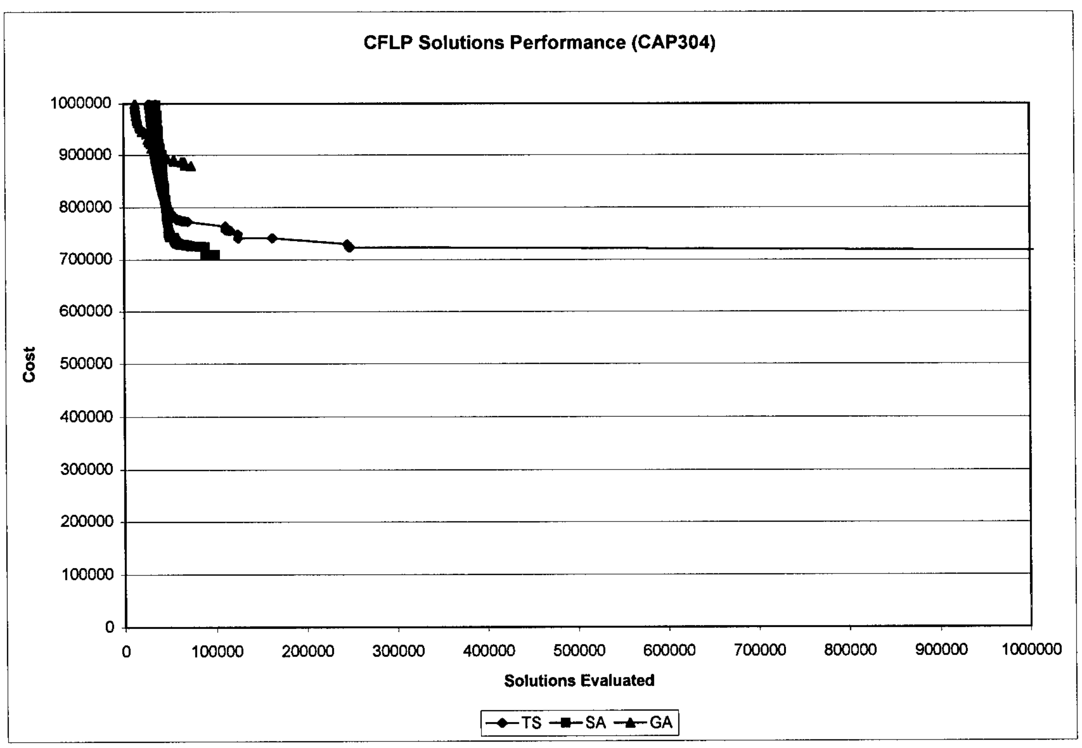

Figure 167. CFLP Solutions Performance Close-up (CAP304)

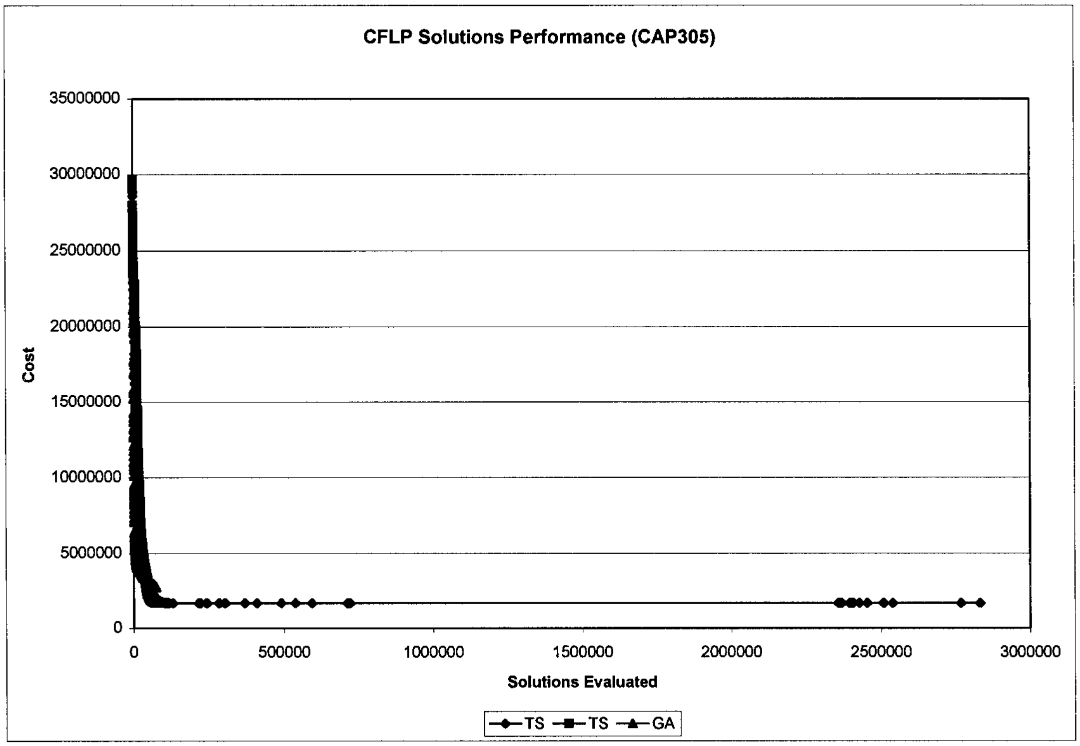

Figure 168. CFLP Solutions Performance (CAP305)

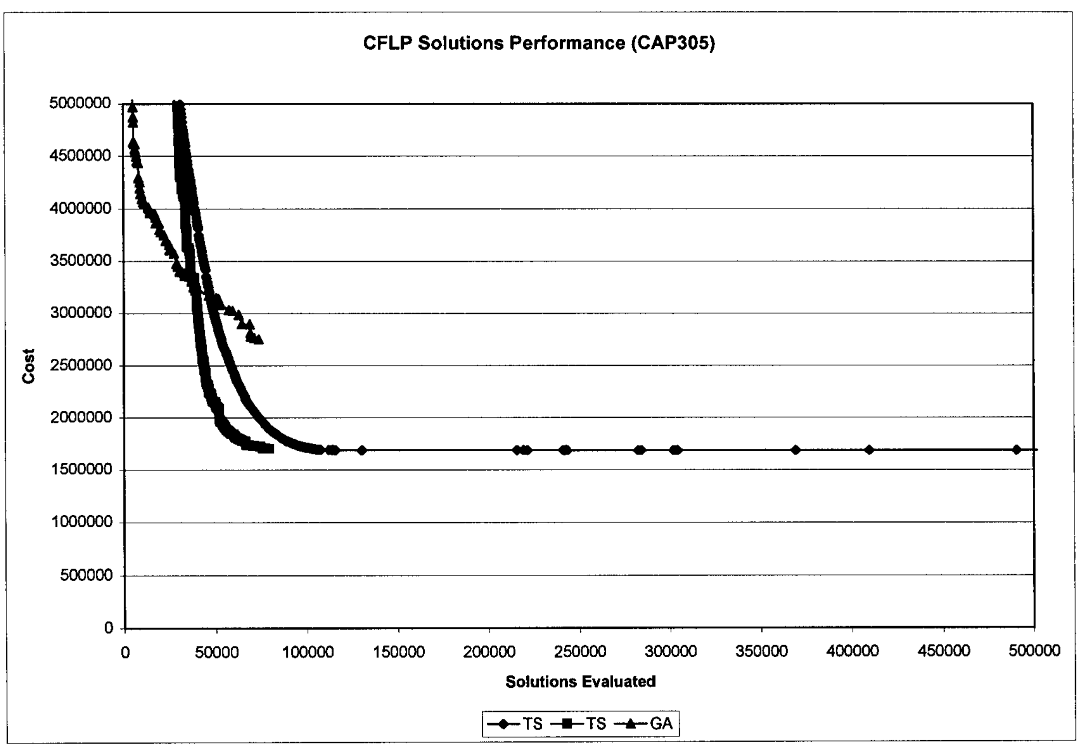

Figure 169. CFLP Solutions Performance Close-up (CAP305)

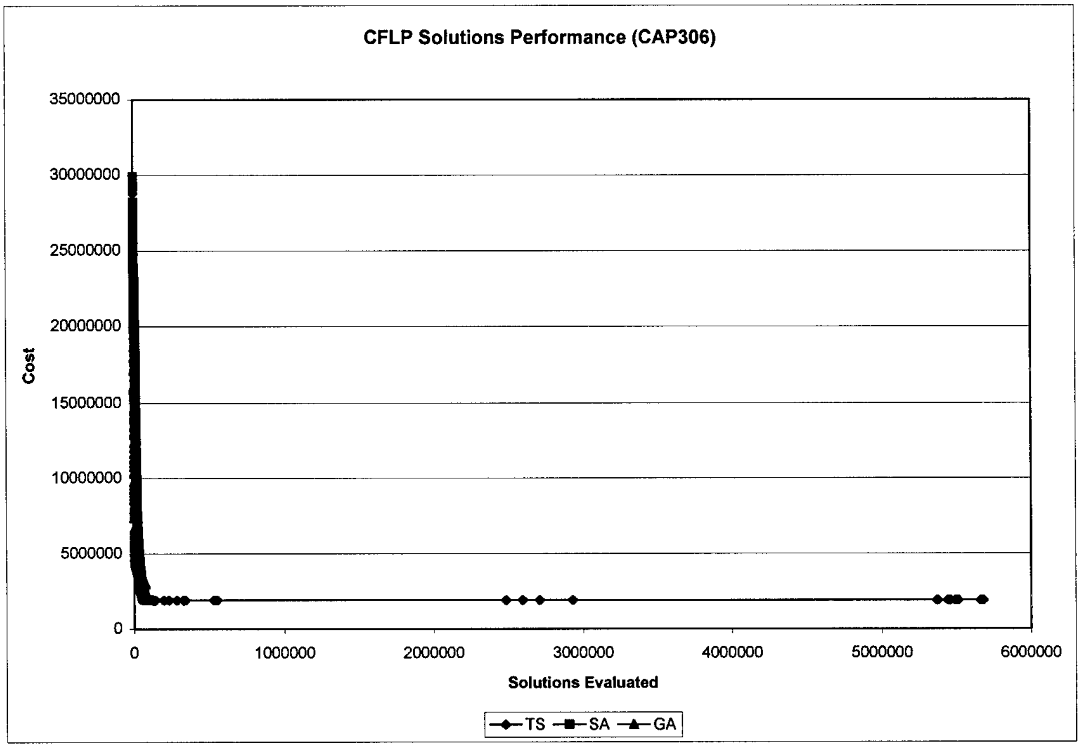

Figure 170. CFLP Solutions Performance (CAP306)

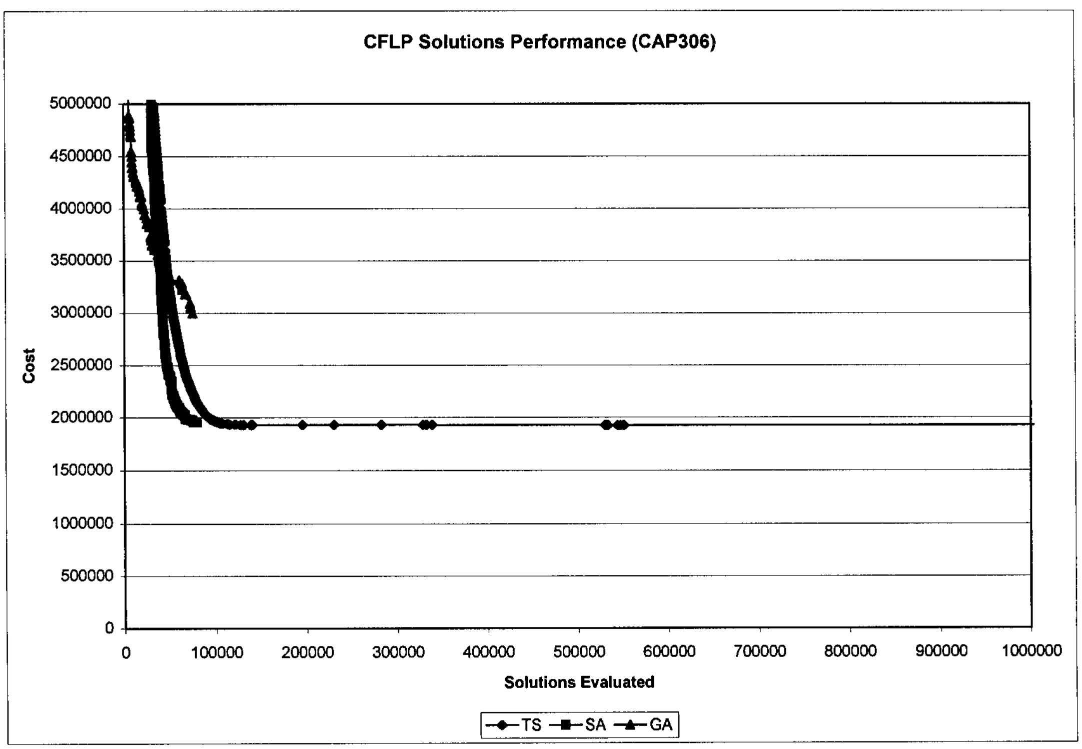

Figure 171. CFLP Solutions Performance Close-up (CAP306)

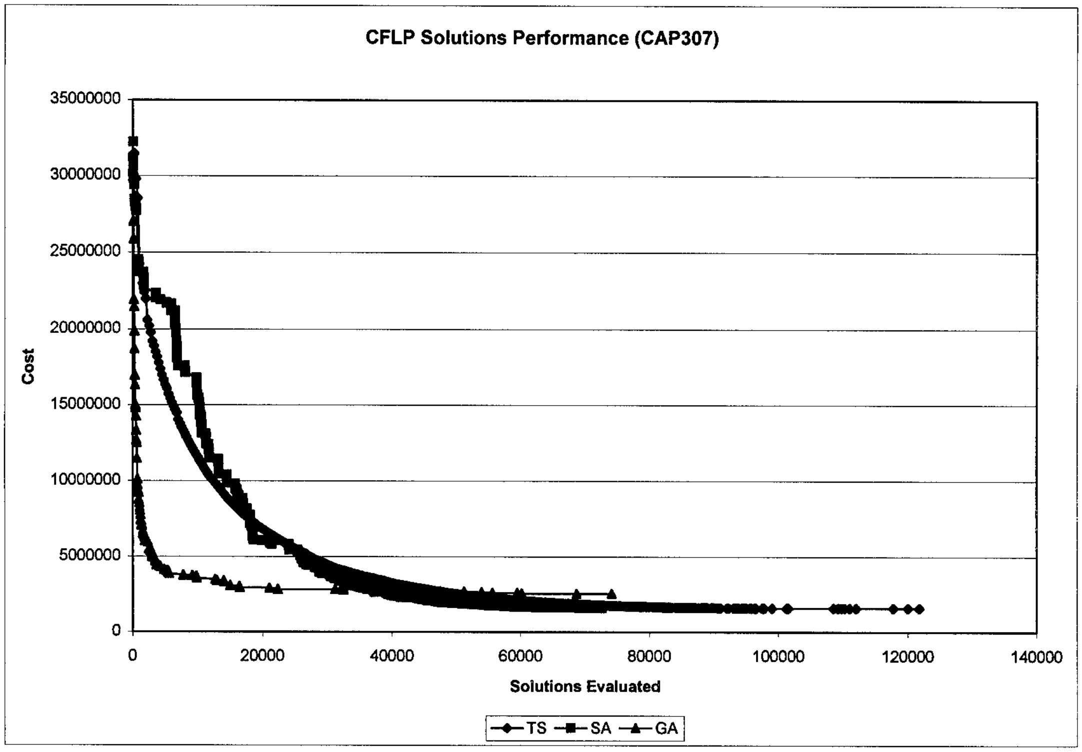

Figure 172. CFLP Solutions Performance (CAP307)

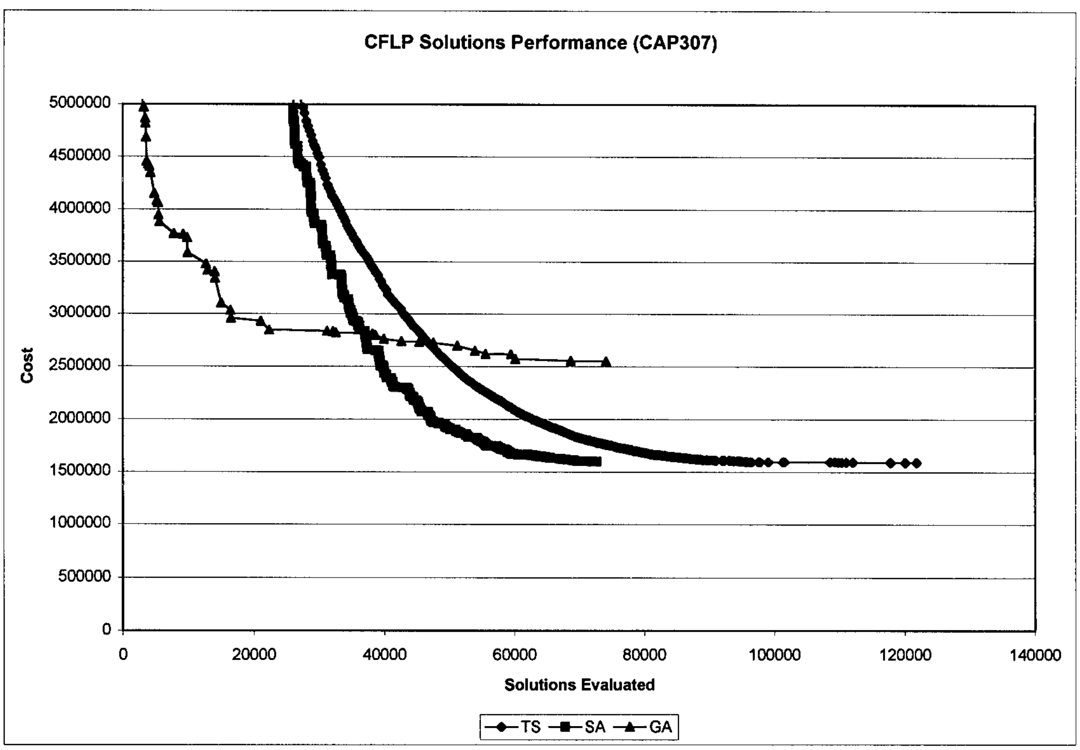

Figure 173. CFLP Solutions Performance Close-up (CAP307)

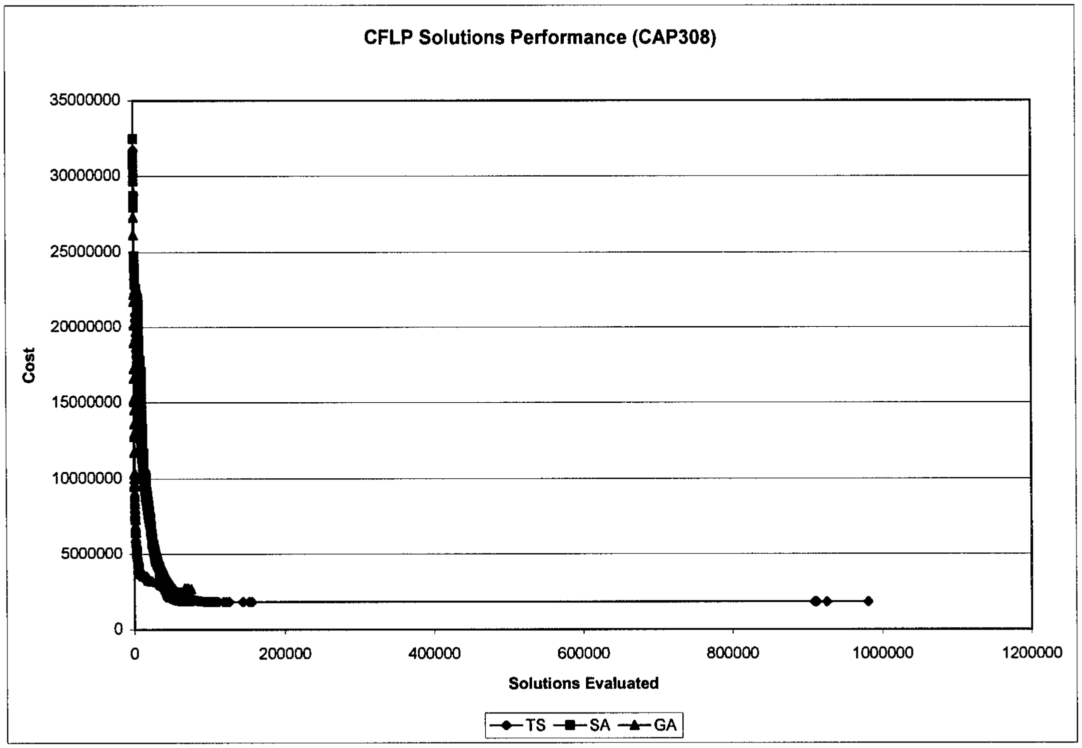

Figure 174. CFLP Solutions Performance (CAP308)

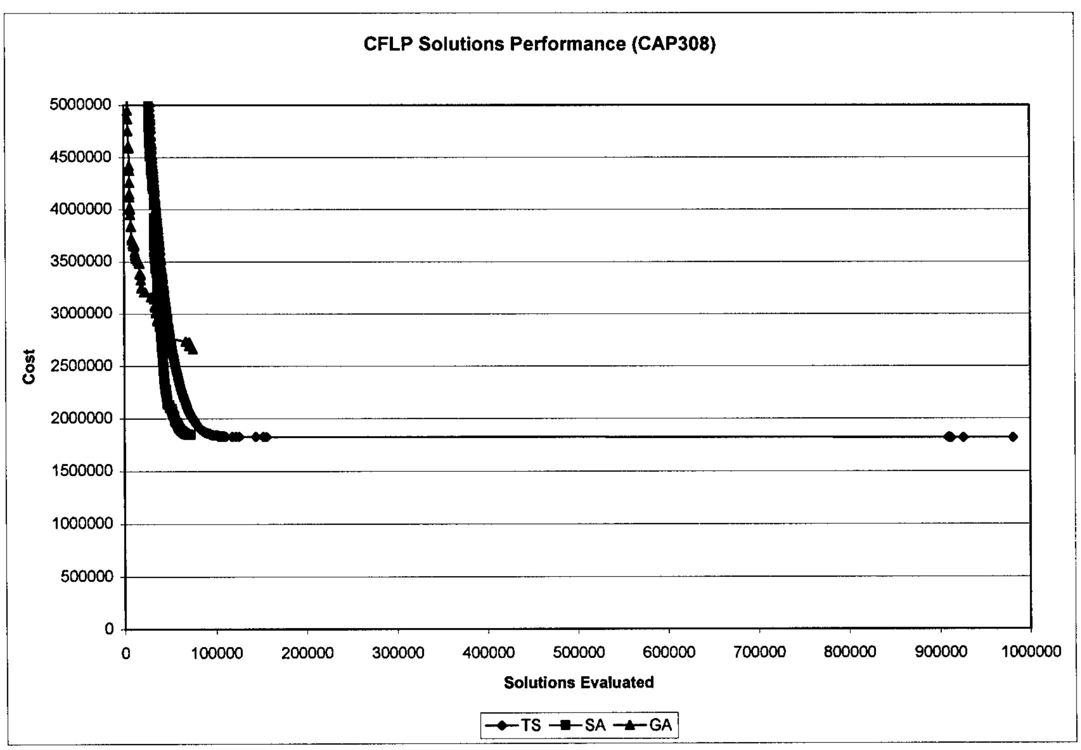

Figure 175. CFLP Solutions Performance Close-up (CAP308)

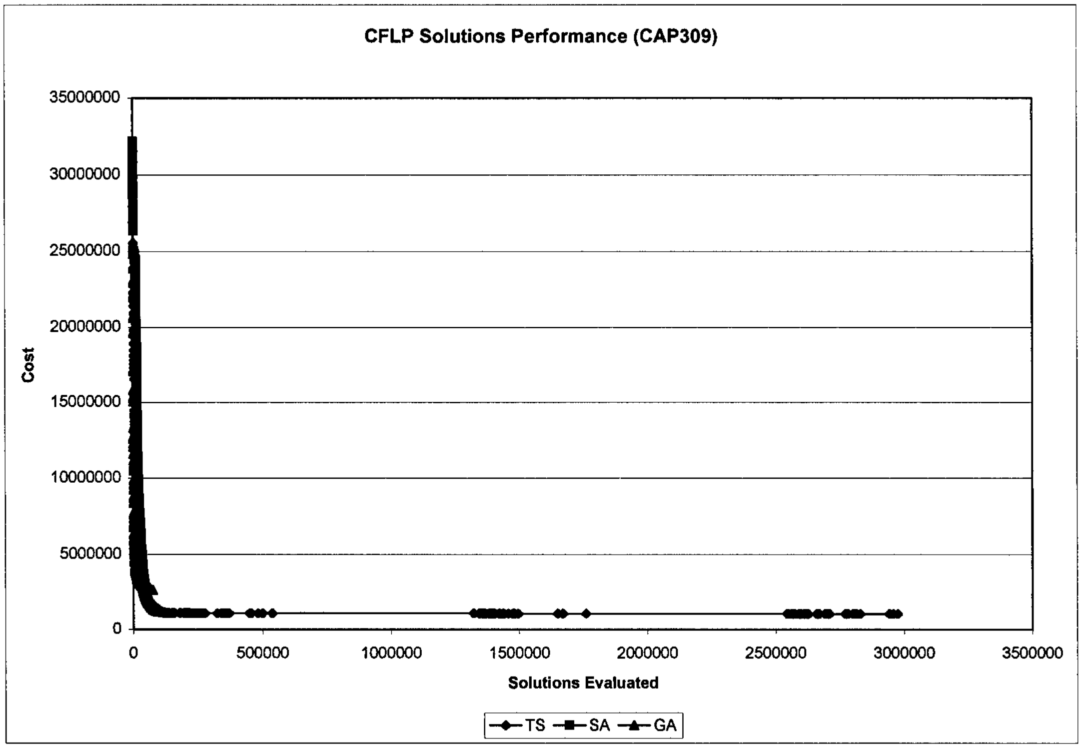

Figure 176. CFLP Solutions Performance (CAP309)

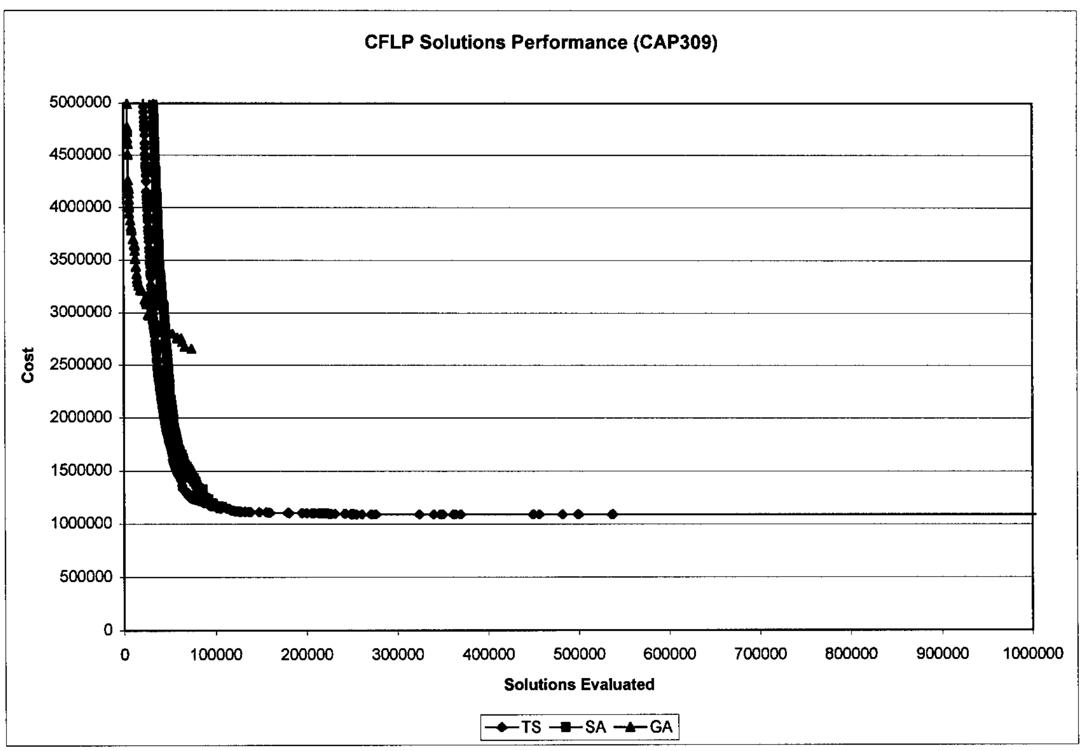

Figure 177. CFLP Solutions Performance Close-up (CAP309)

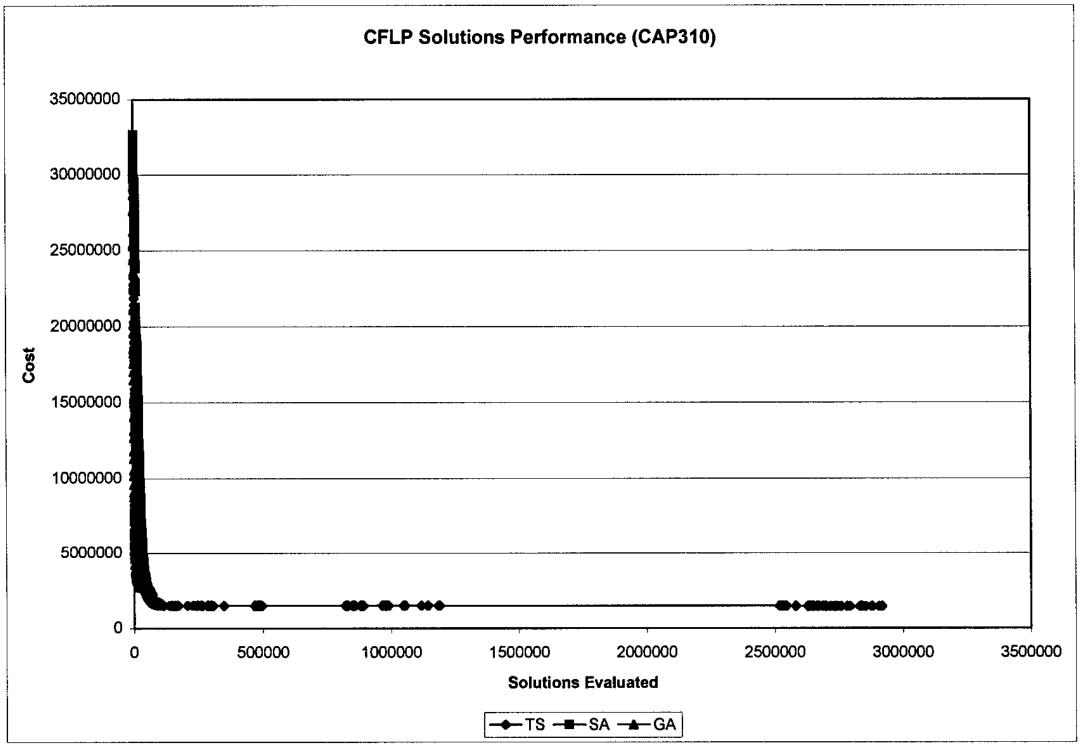

Figure 178. CFLP Solutions Performance (CAP310)

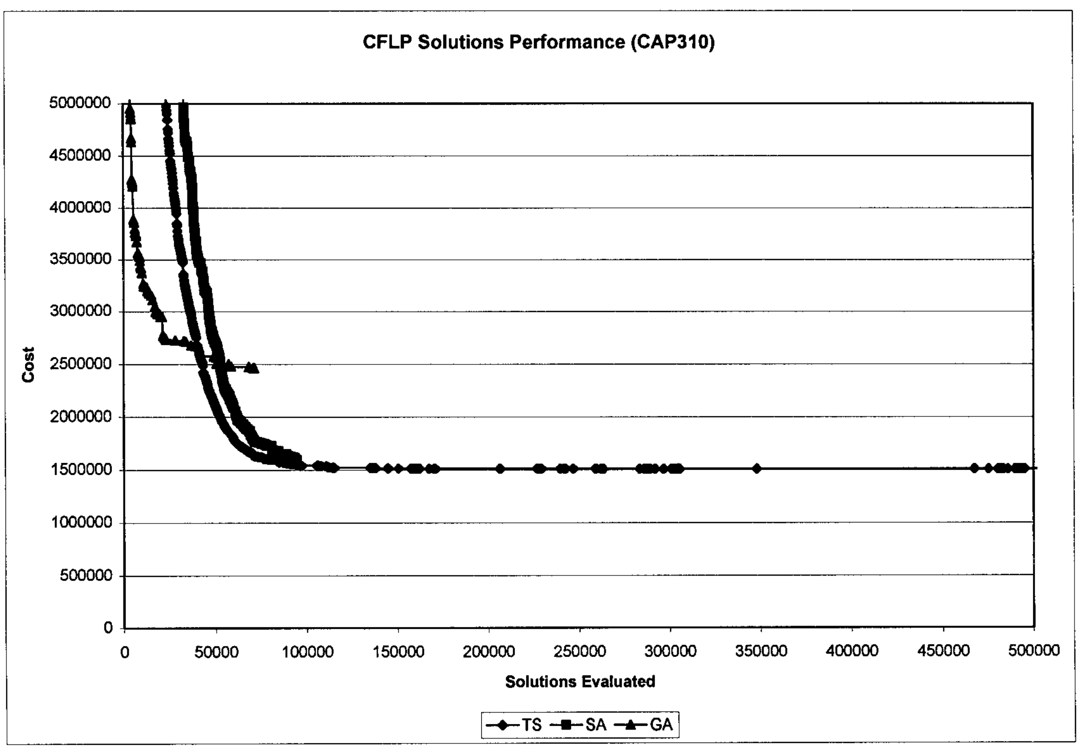

Figure 179. CFLP Solutions Performance Close-up (CAP310)

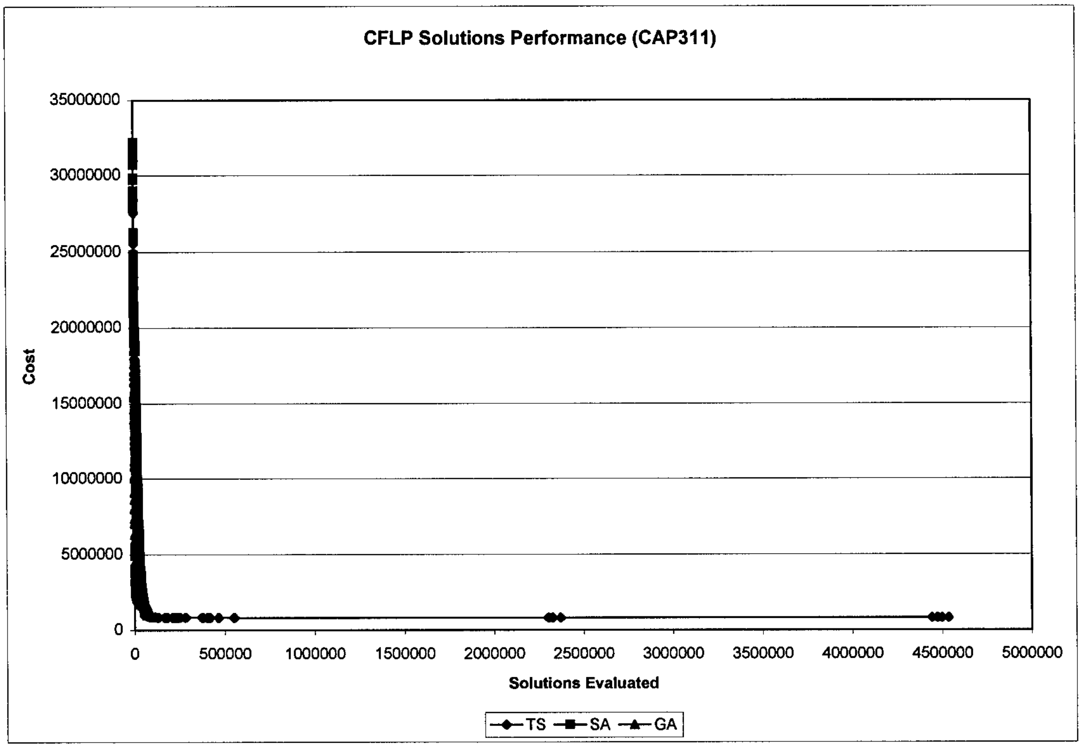

Figure 180. CFLP Solutions Performance (CAP311)

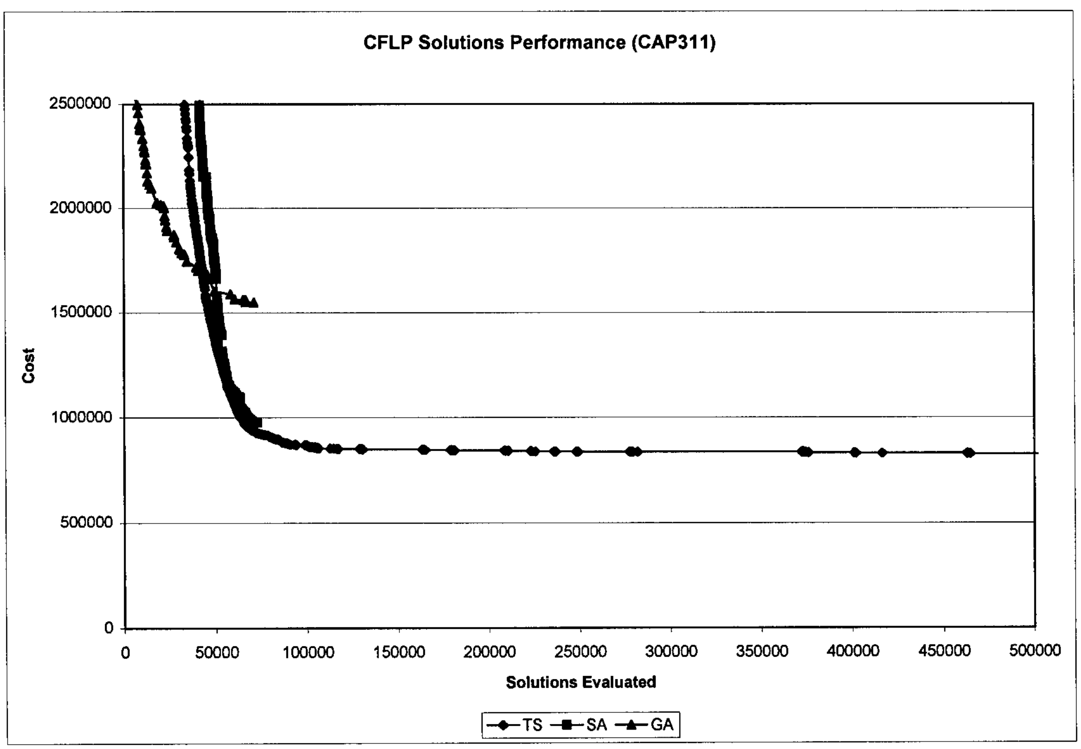

Figure 181. CFLP Solutions Performance Close-up (CAP311)

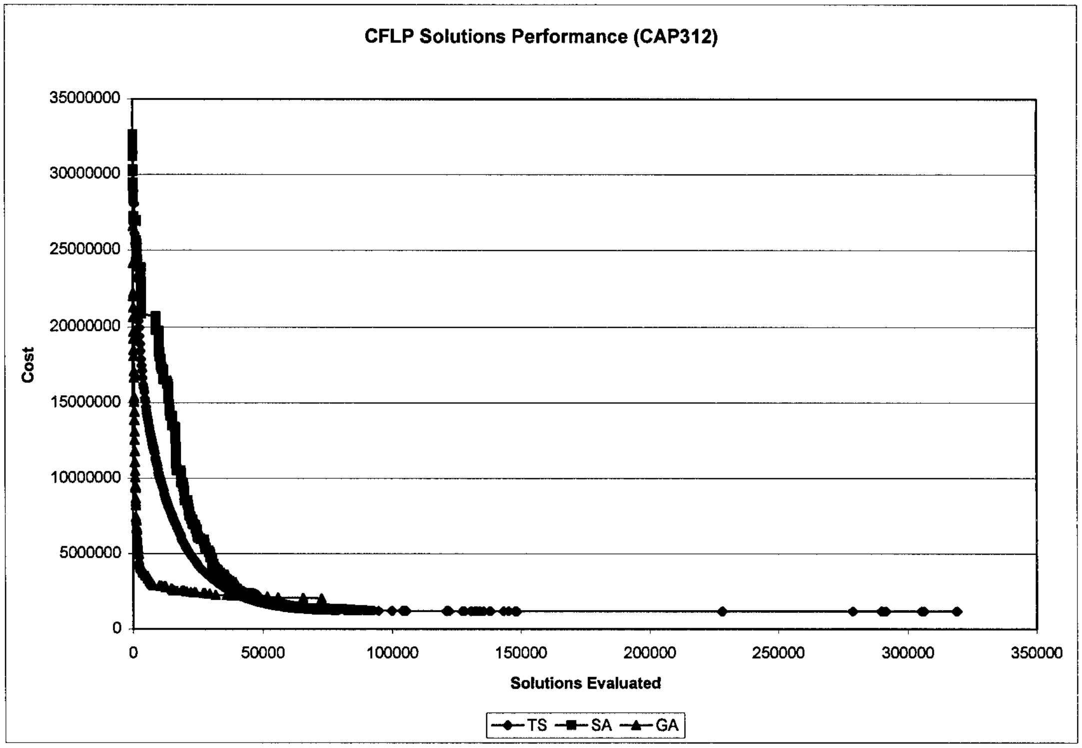

Figure 182. CFLP Solutions Performance (CAP312)

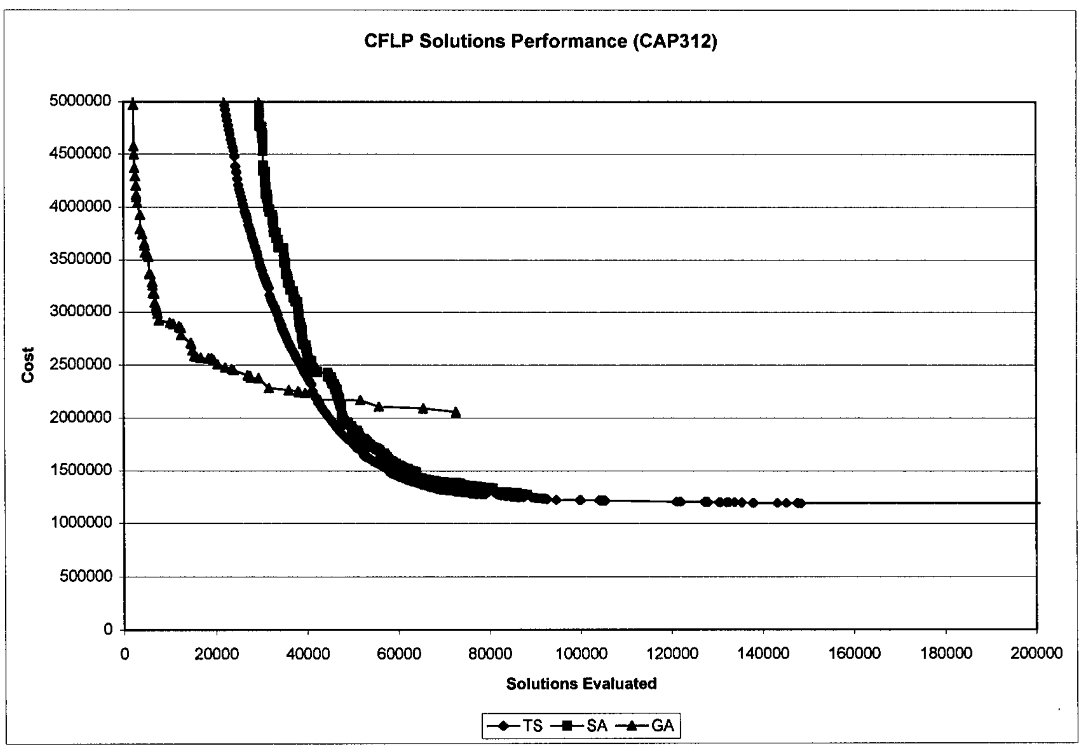

Figure 183. CFLP Solutions Performance Close-up (CAP312)

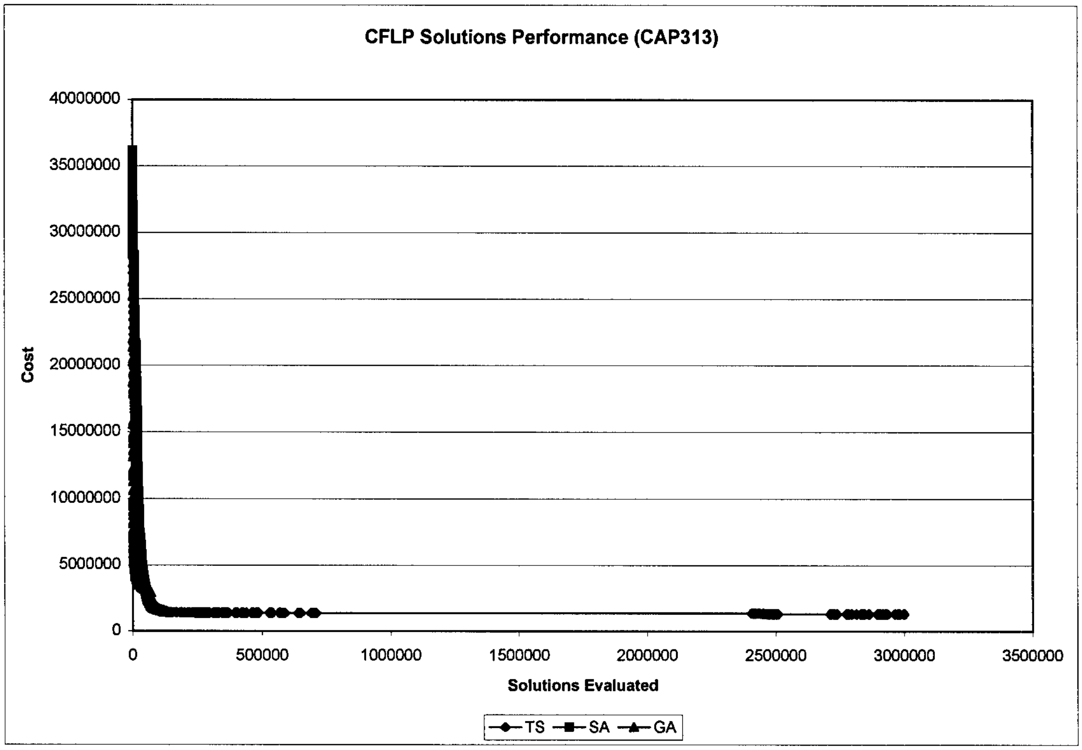

Figure 184. CFLP Solutions Performance (CAP313)

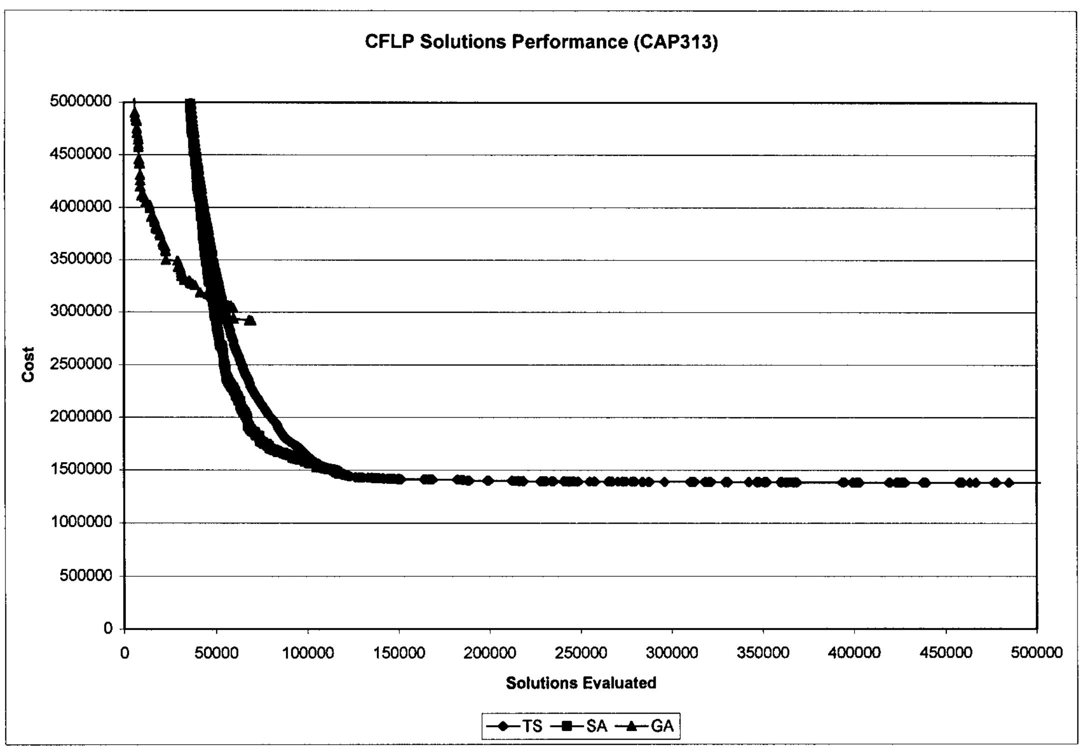

Figure 185. CFLP Solutions Performance Close-up (CAP313)

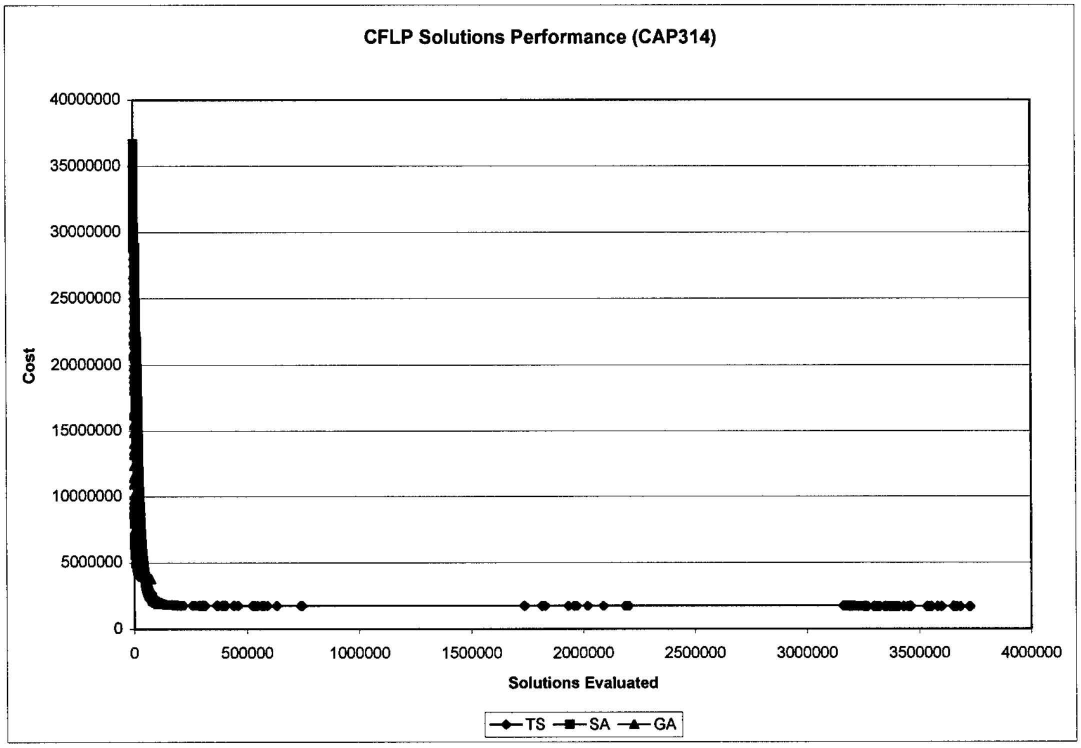

Figure 186. CFLP Solutions Performance (CAP314)

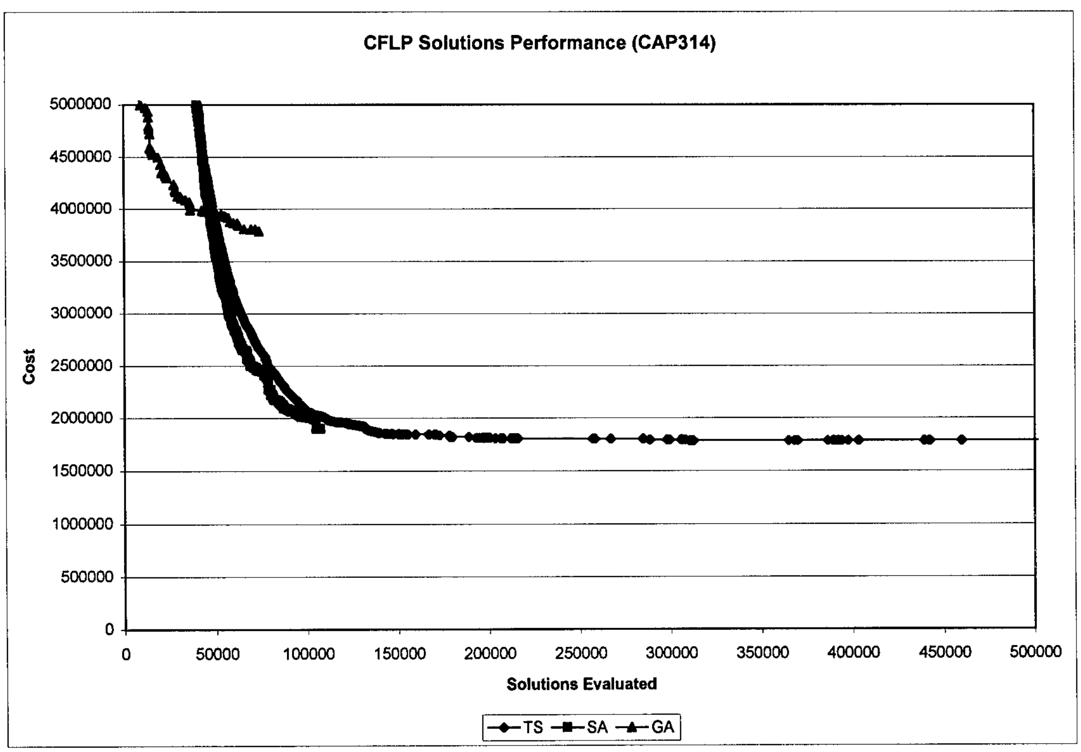

Figure 187. CFLP Solutions Performance Close-up (CAP314)

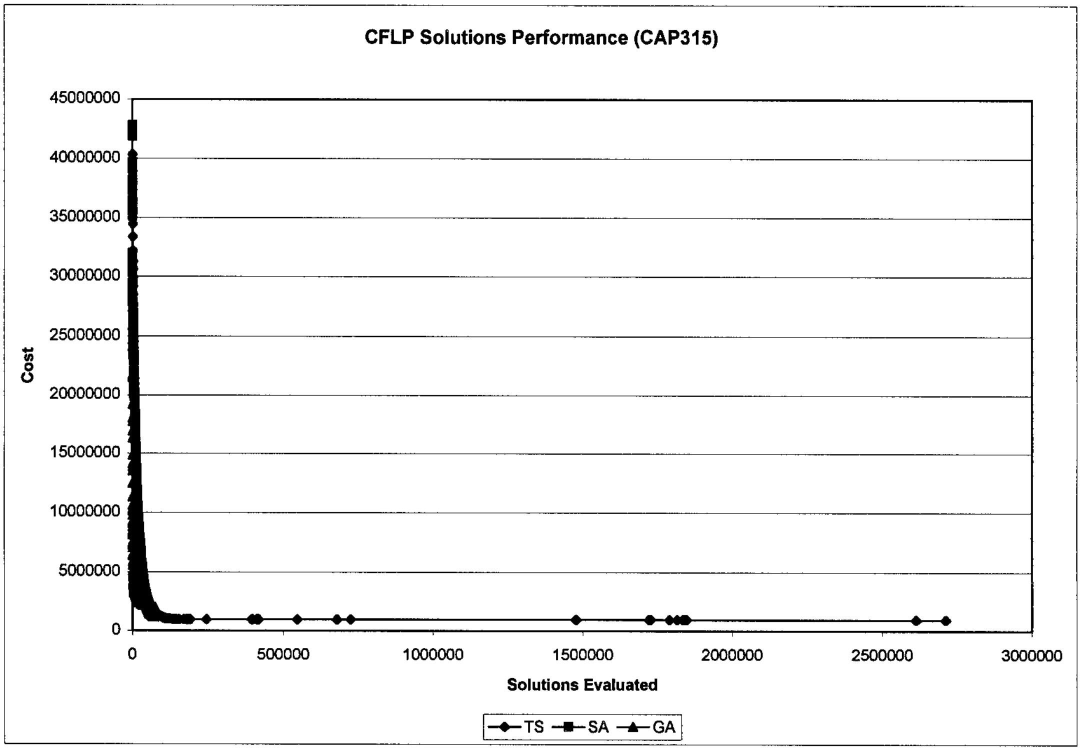

Figure 188. CFLP Solutions Performance (CAP315)

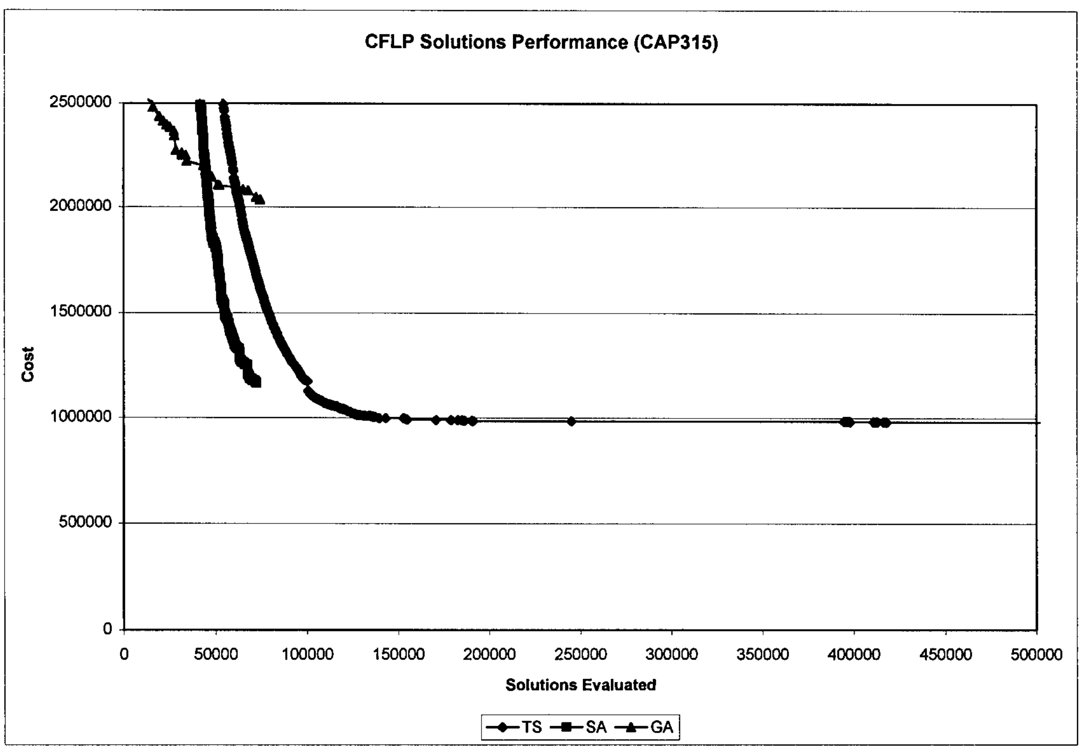

Figure 189. CFLP Solutions Performance Close-up (CAP315)

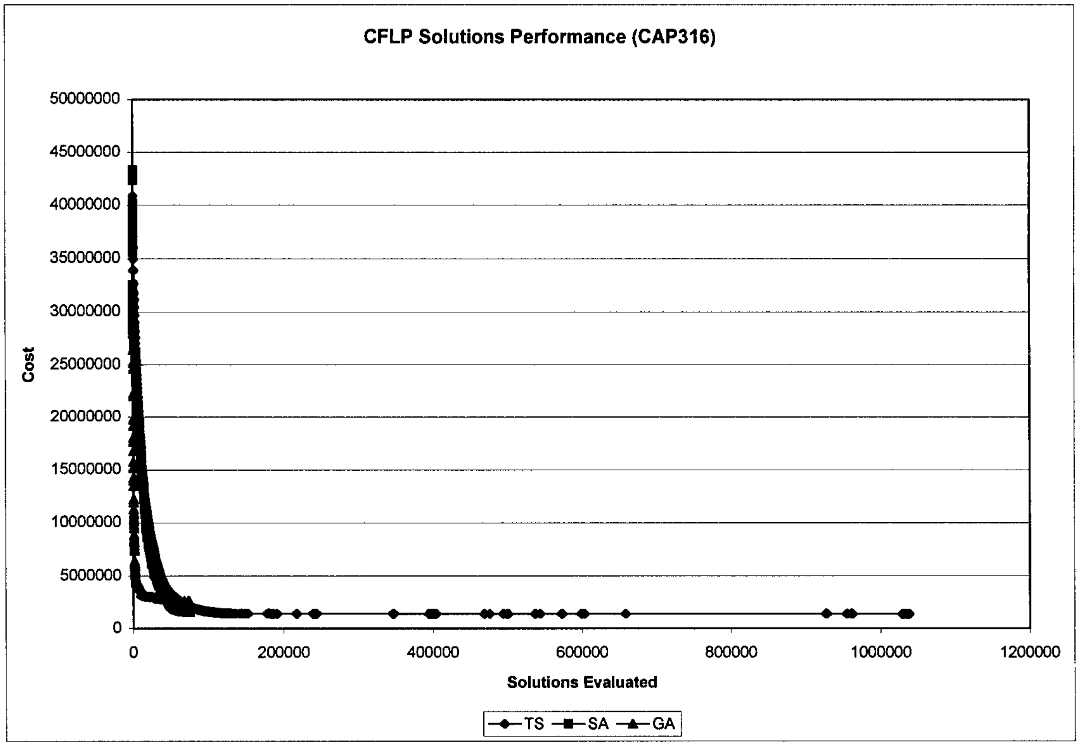

Figure 190. CFLP Solutions Performance (CAP316)

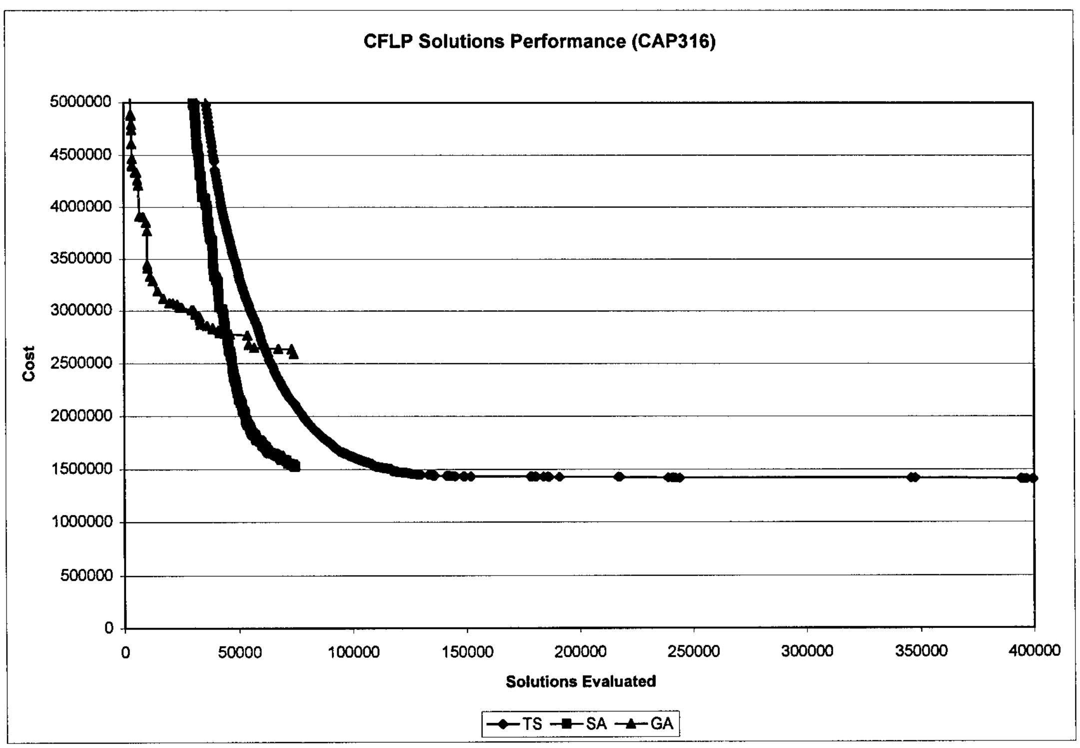

Figure 191. CFLP Solutions Performance Close-up (CAP316)

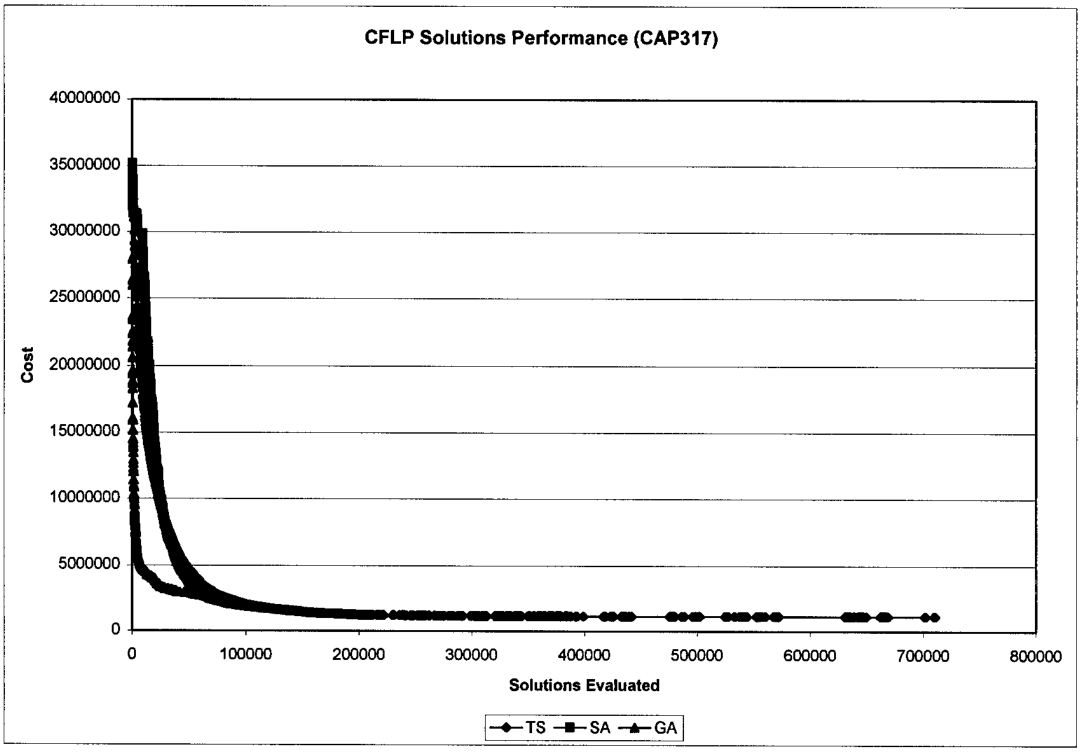

Figure 192. CFLP Solutions Performance (CAP317)

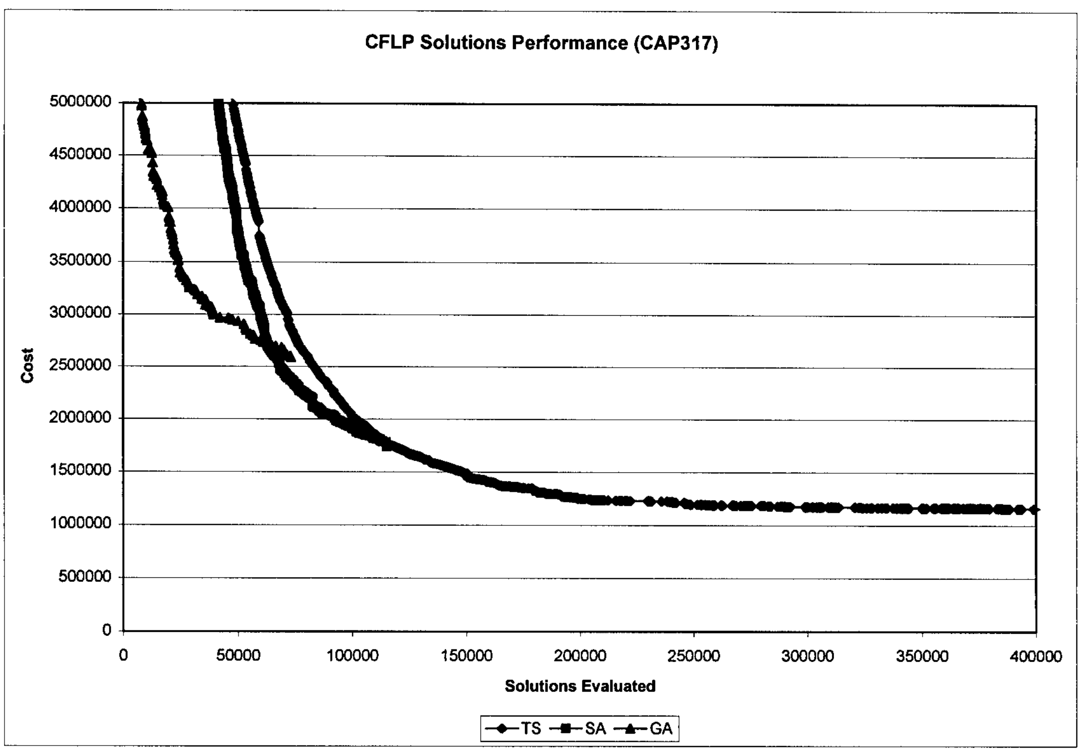

Figure 193. CFLP Solutions Performance Close-up (CAP317)

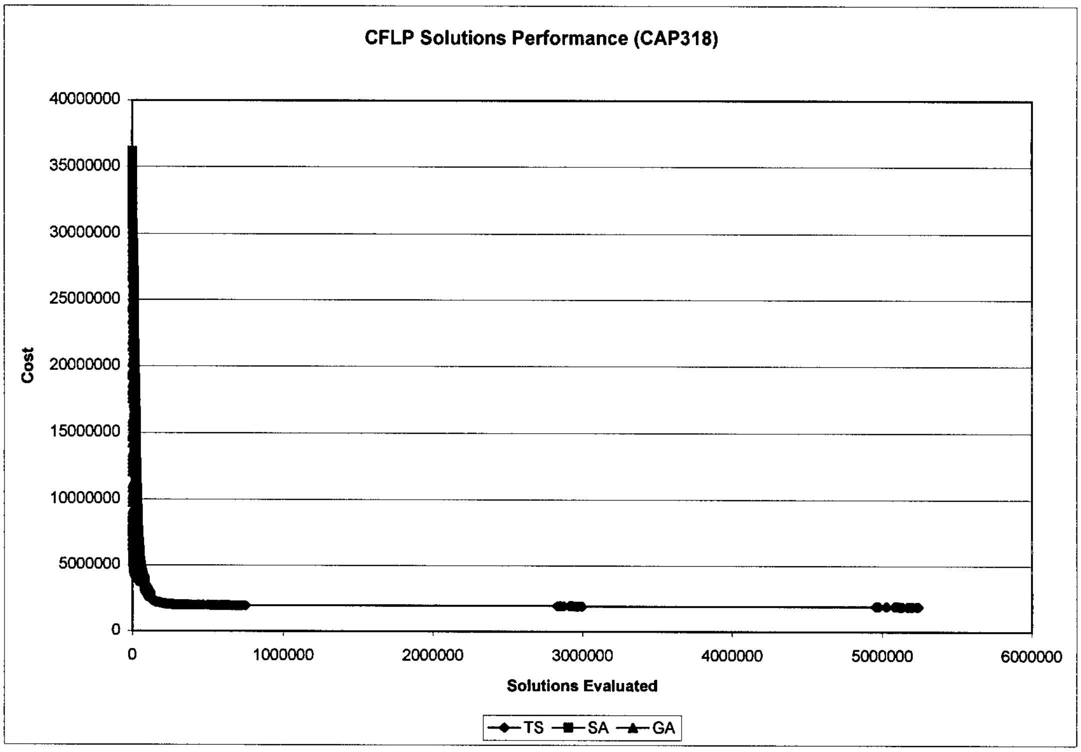

Figure 194. CFLP Solutions Performance (CAP318)

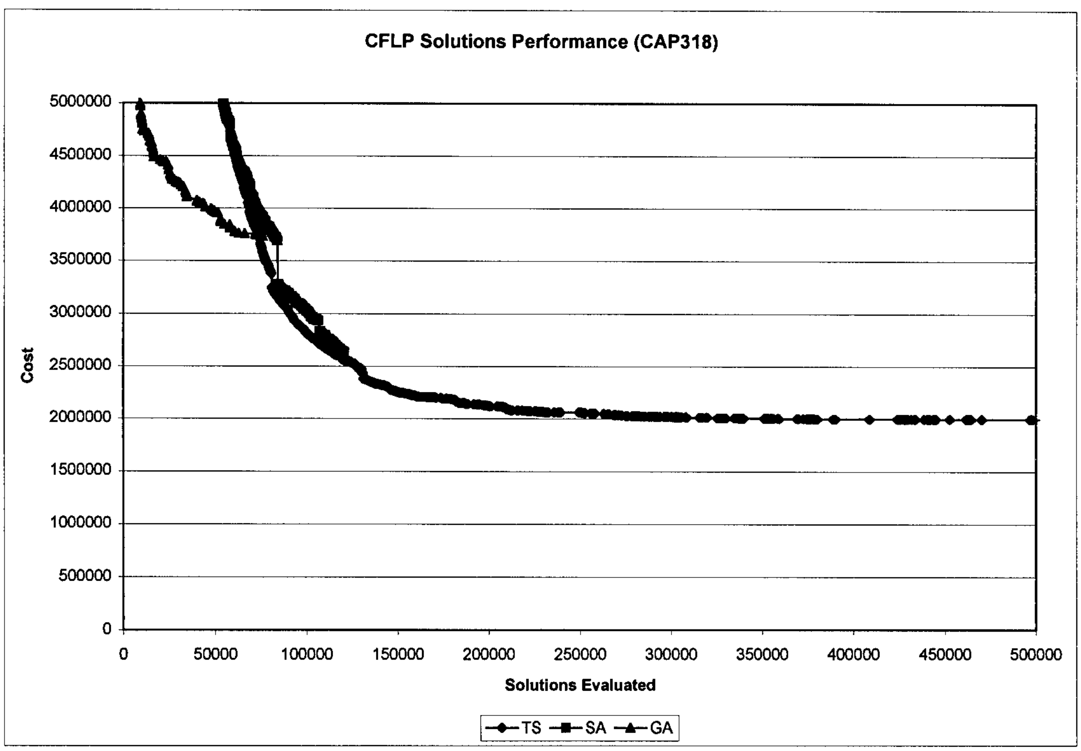

Figure 195. CFLP Solutions Performance Close-up (CAP318)

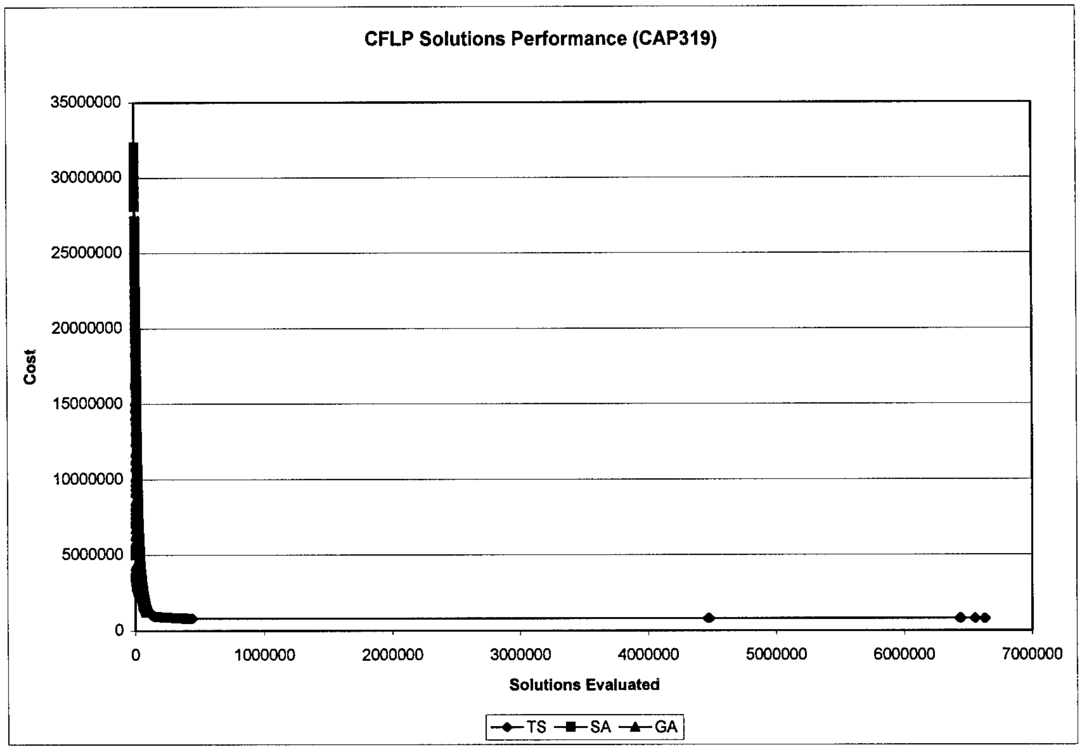

Figure 196. CFLP Solutions Performance (CAP319)

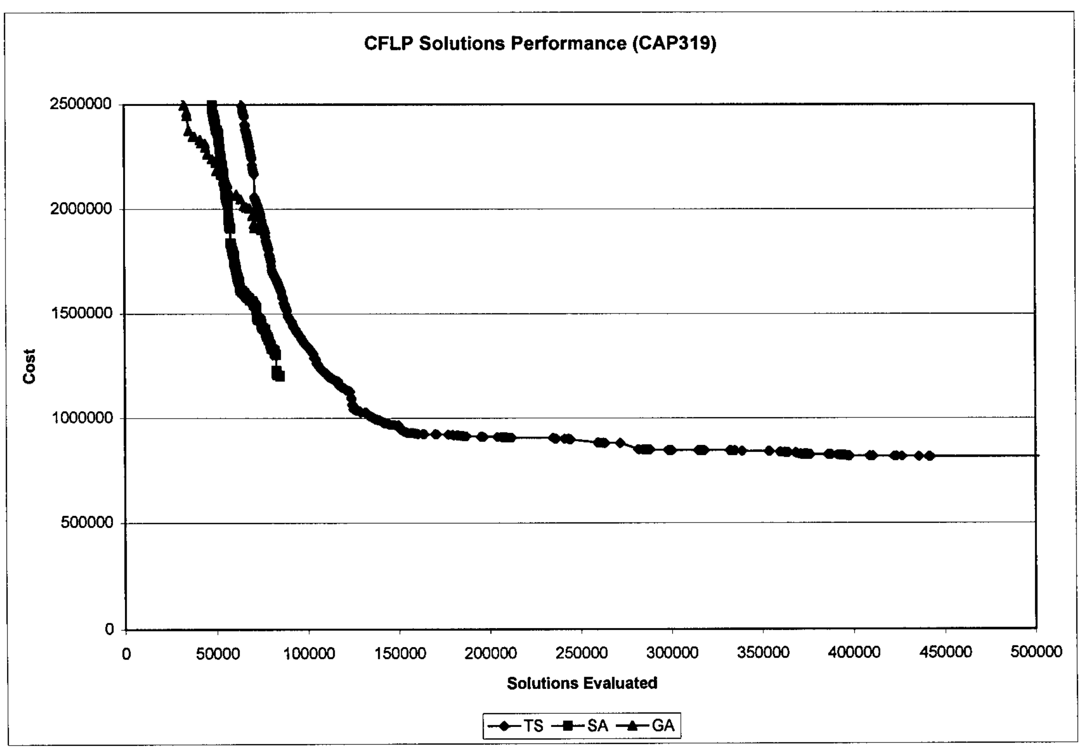

Figure 197. CFLP Solutions Performance Close-up (CAP319)

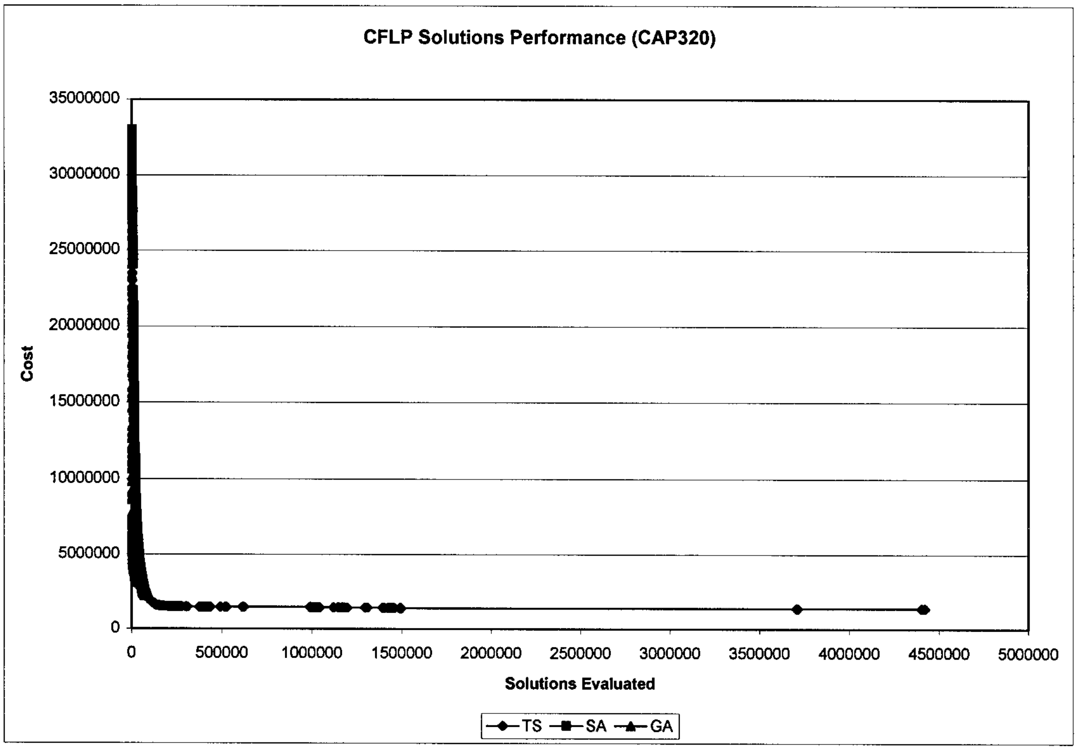

Figure 198. CFLP Solutions Performance (CAP320)

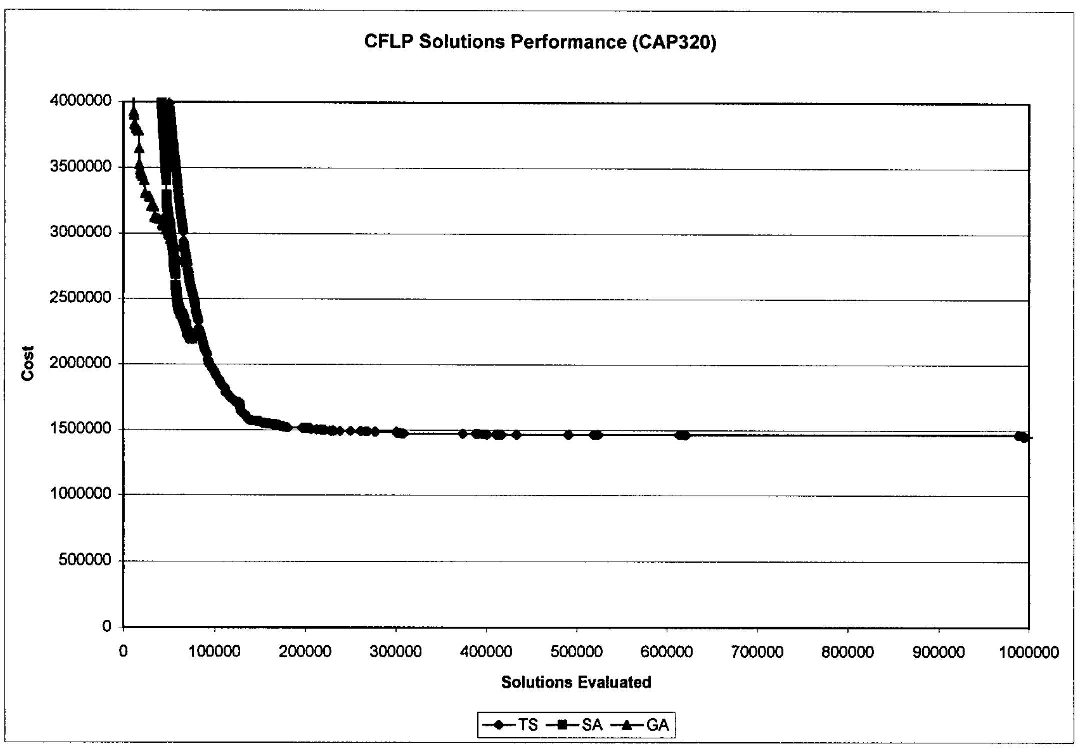

Figure 199. CFLP Solutions Performance Close-up (CAP320)

APPENDIX G

MP-FLP TIME PERFORMANCE CHARTS

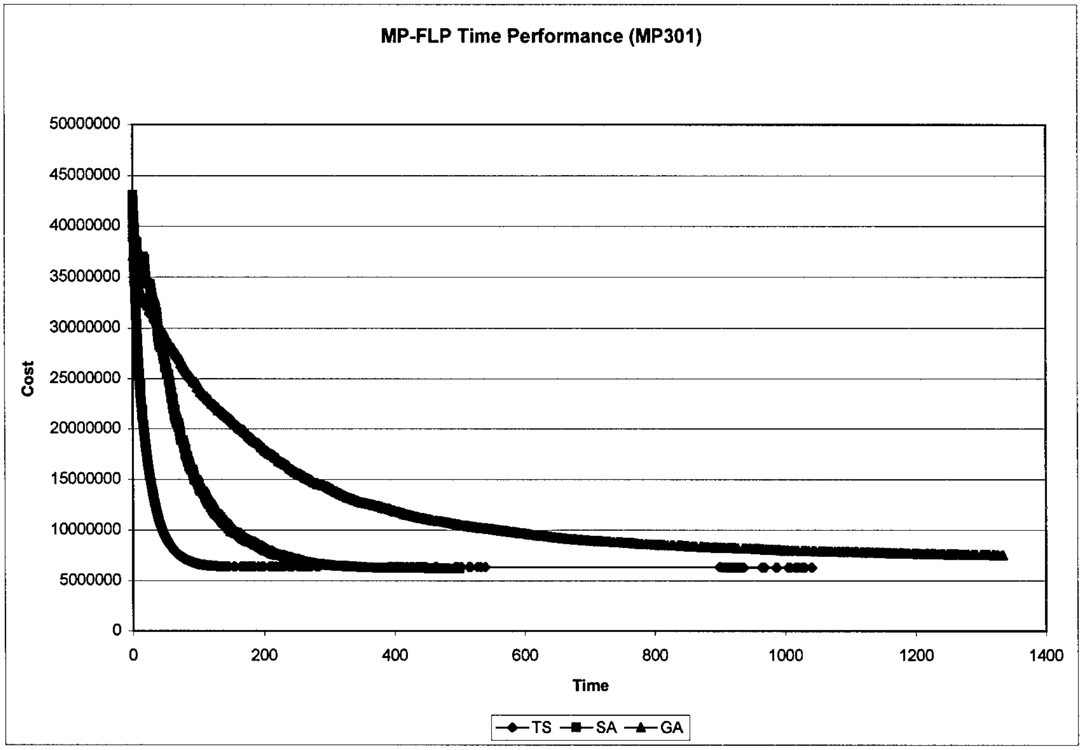

Figure 200. MP-FLP Time Performance (MP301)

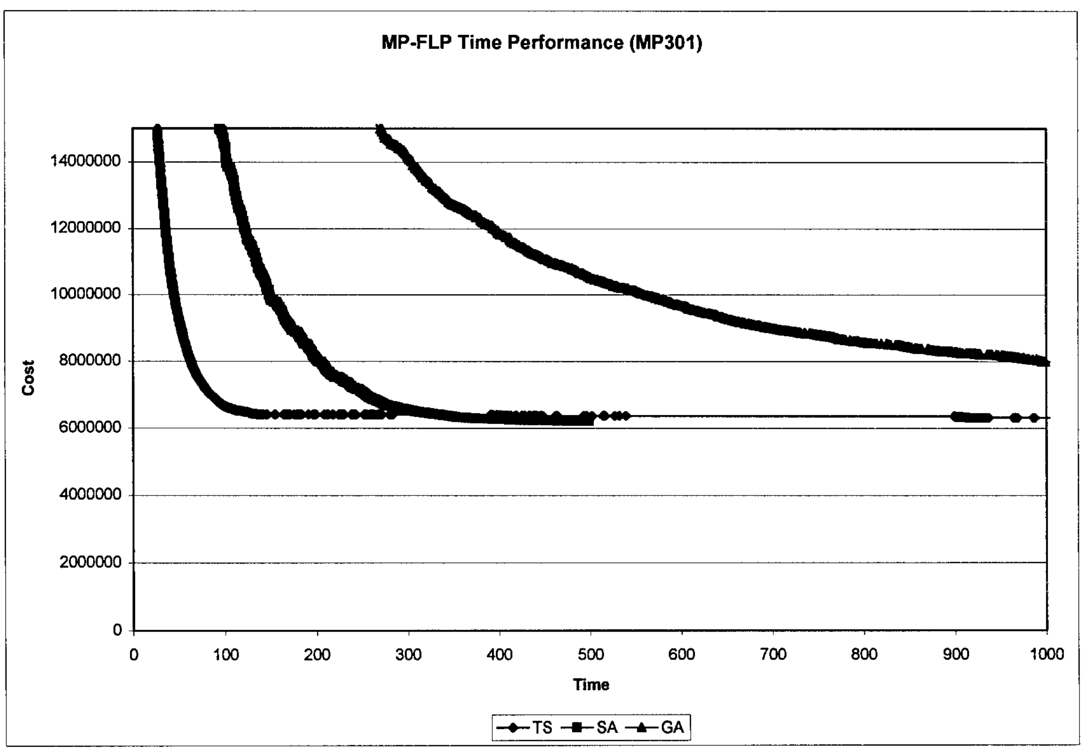

Figure 201. MP-FLP Time Performance Close-up (MP301)

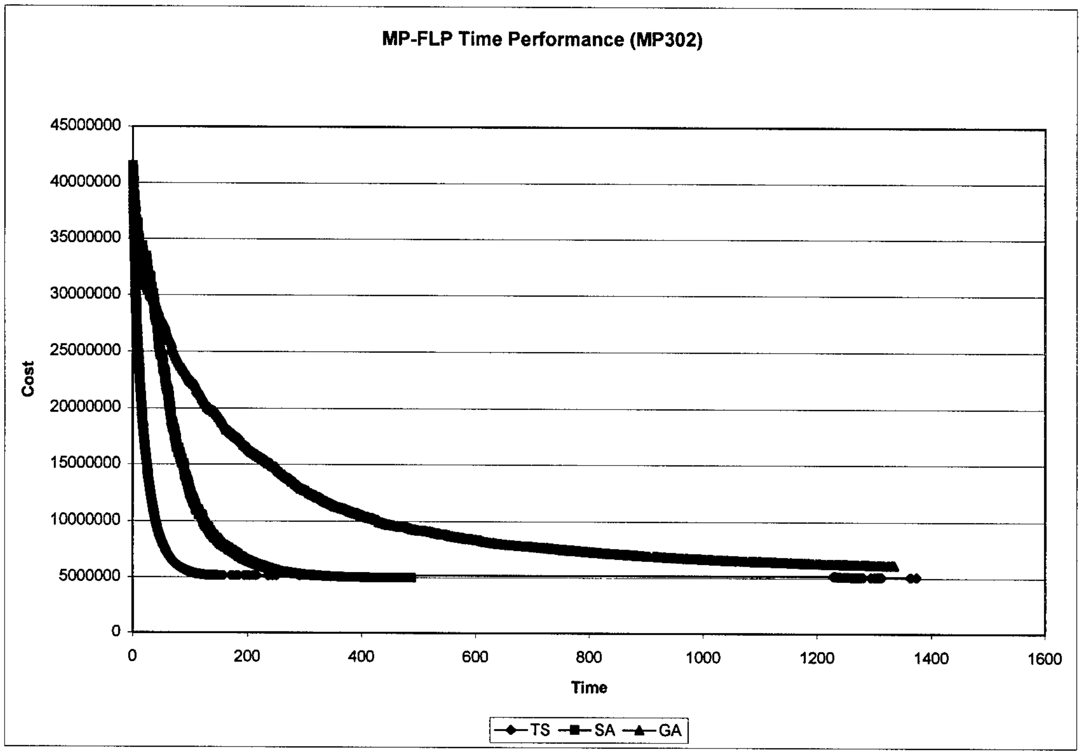

Figure 202. MP-FLP Time Performance (MP302)

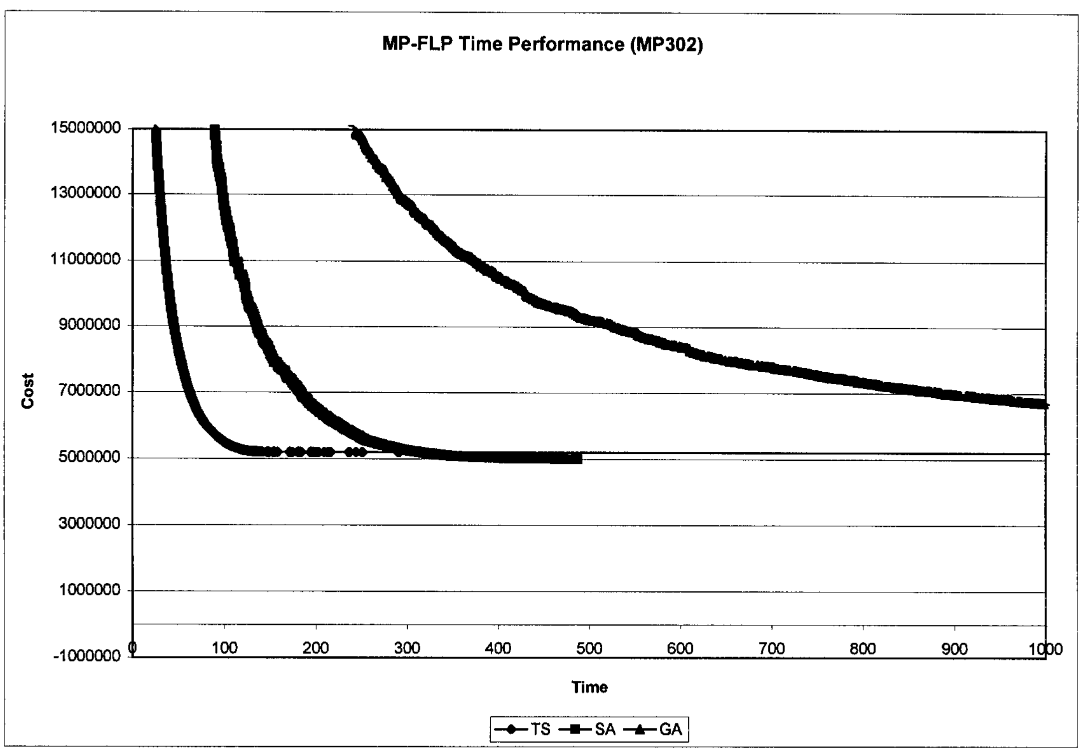

Figure 203. MP-FLP Time Performance Close-up (MP302)

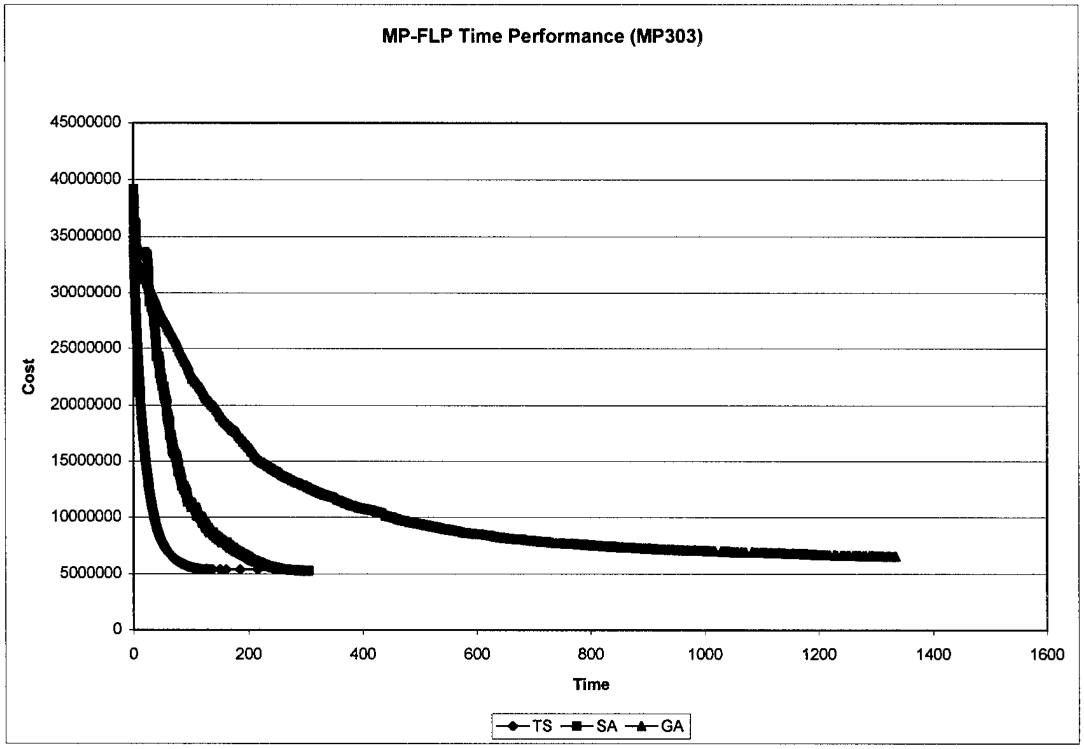

Figure 204. MP-FLP Time Performance (MP303)

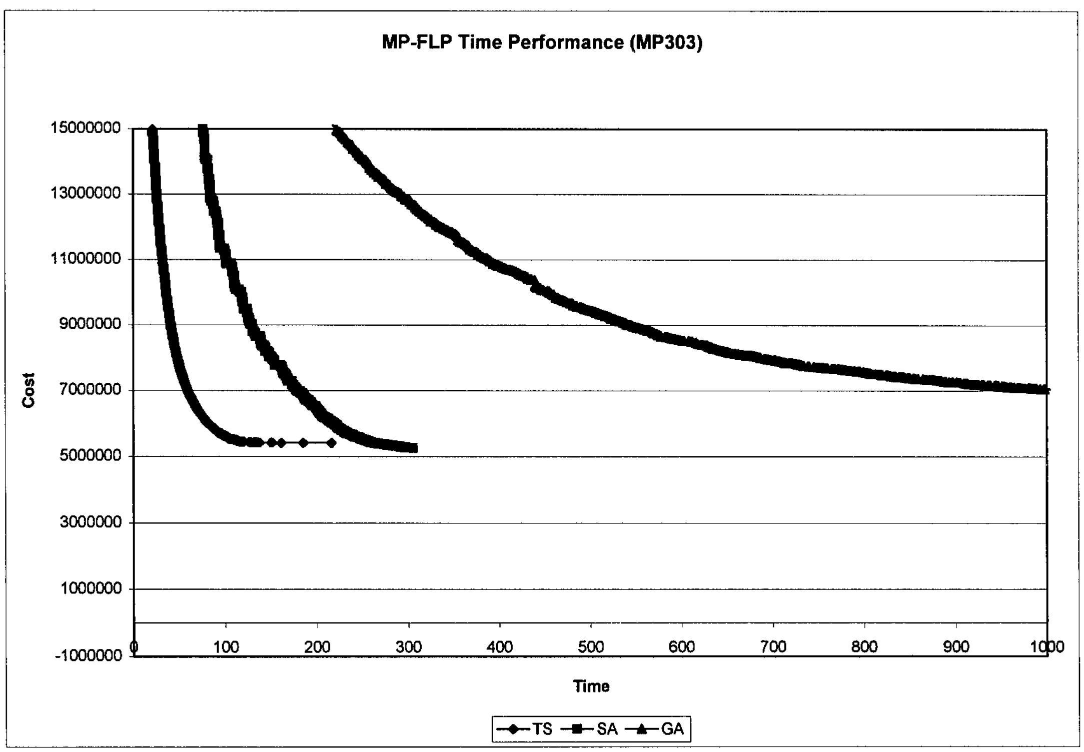

Figure 205. MP-FLP Time Performance Close-up (MP303)

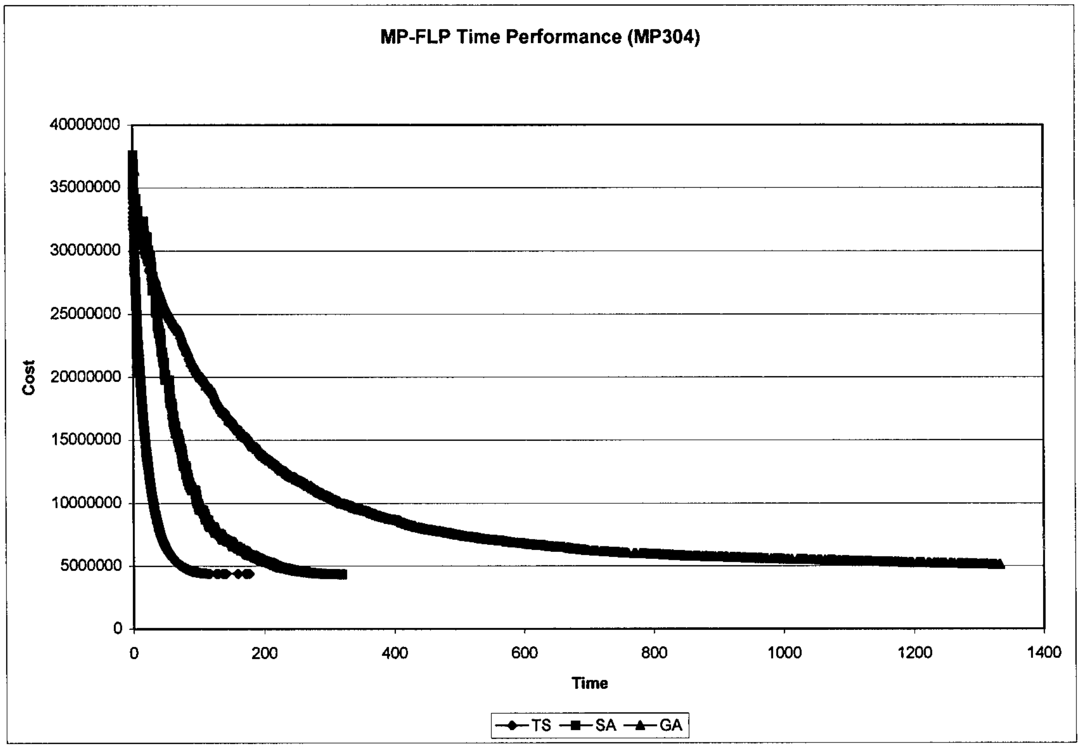

Figure 206. MP-FLP Time Performance (MP304)

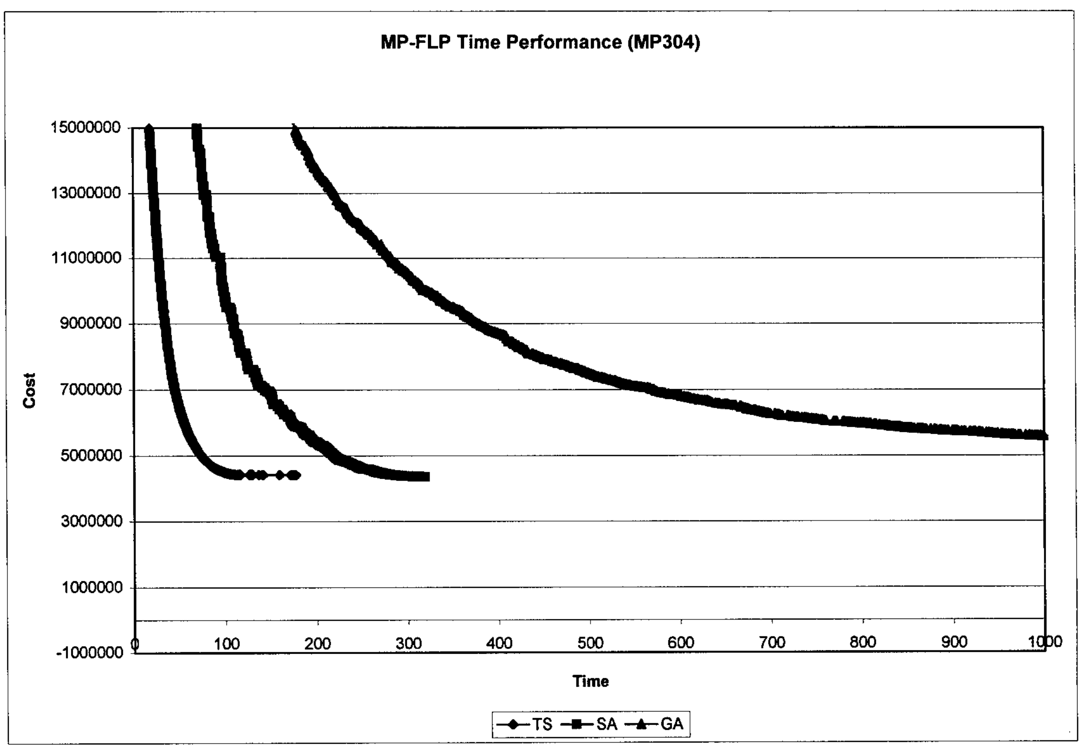

Figure 207. MP-FLP Time Performance Close-up (MP304)

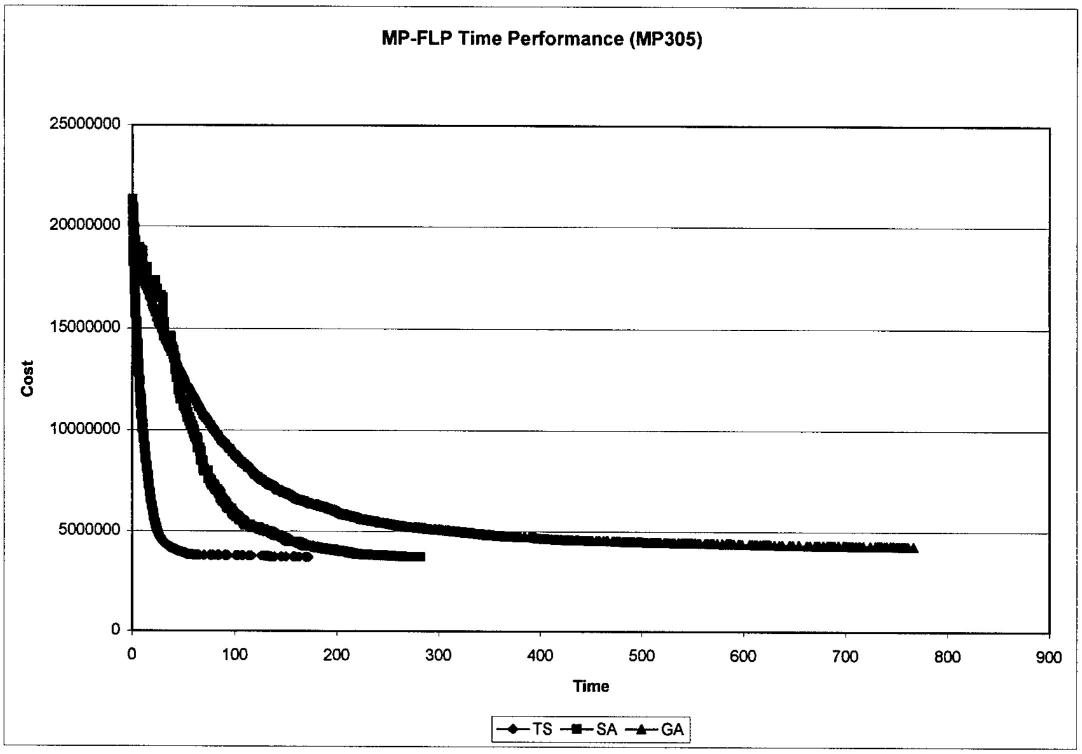

Figure 208. MP-FLP Time Performance (MP305)

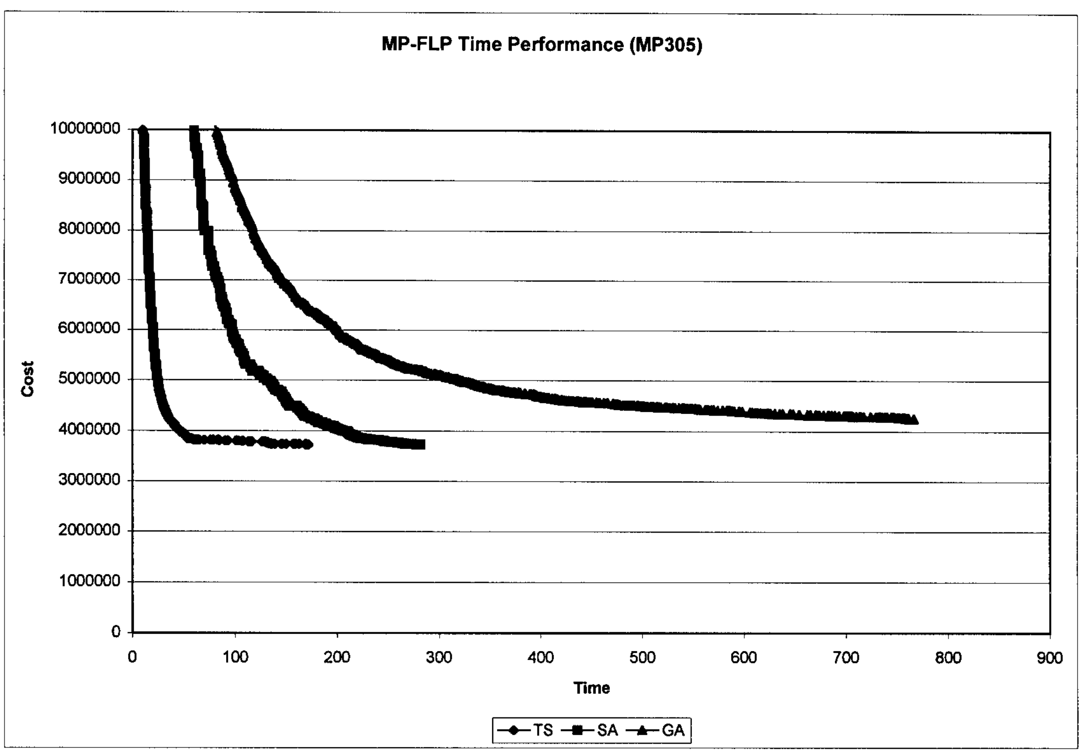

Figure 209. MP-FLP Time Performance Close-up (MP305)

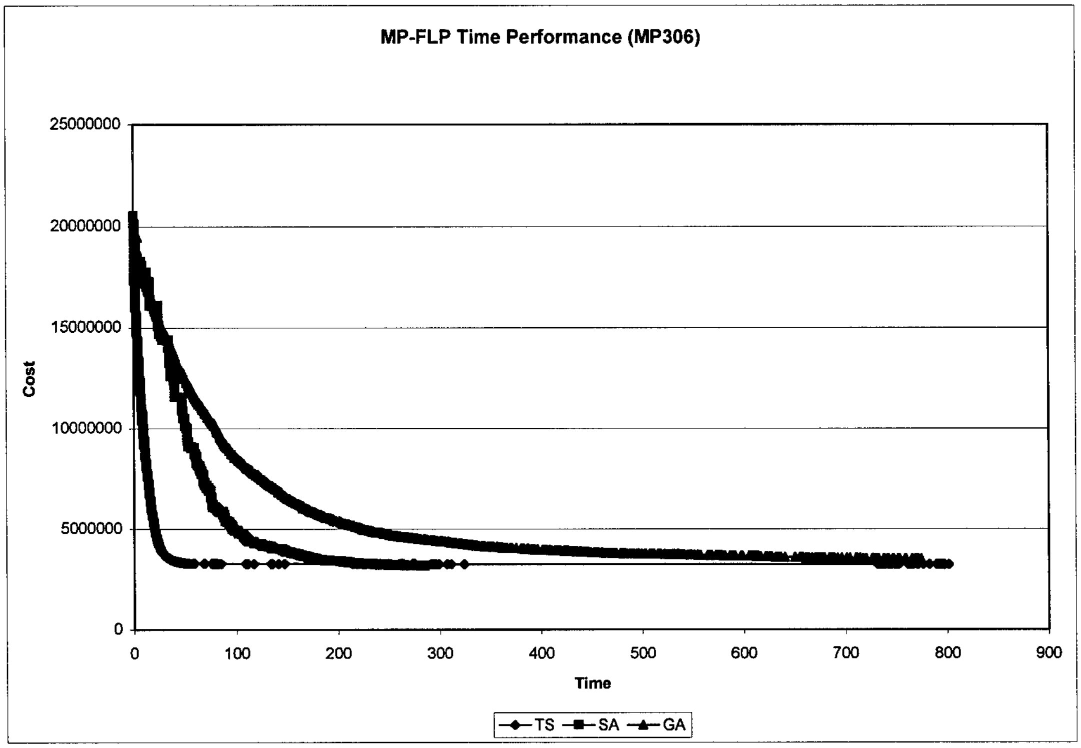

Figure 210. MP-FLP Time Performance (MP306)

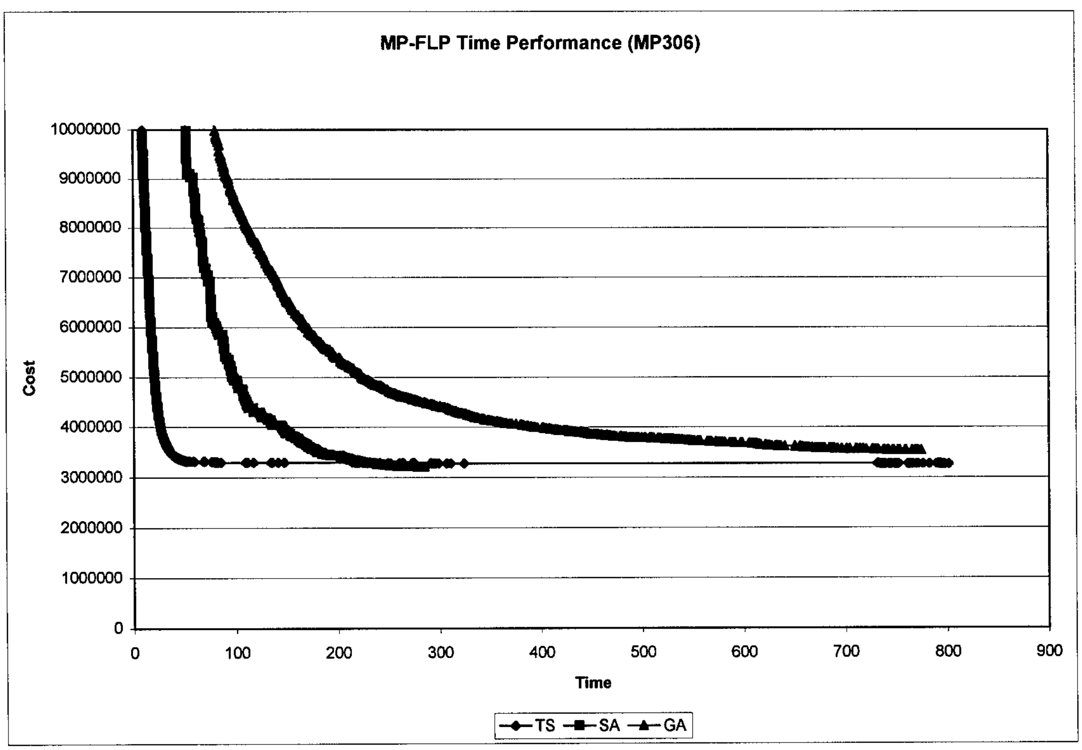

Figure 211. MP-FLP Time Performance Close-up (MP306)

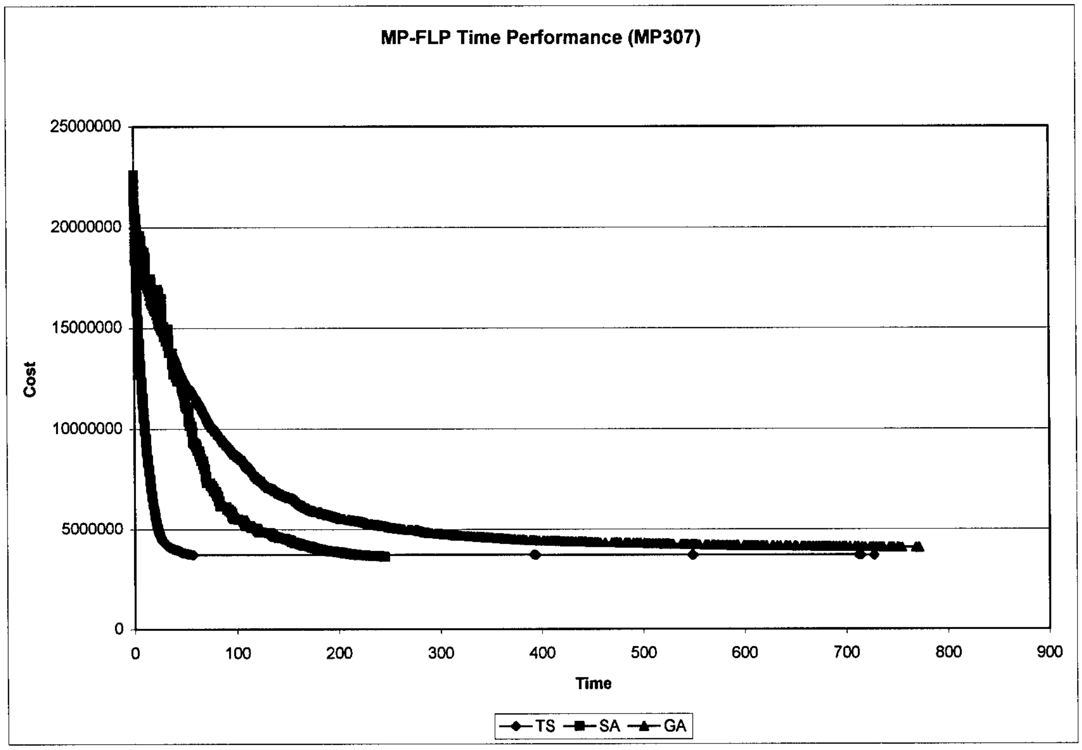

Figure 212. MP-FLP Time Performance (MP307)

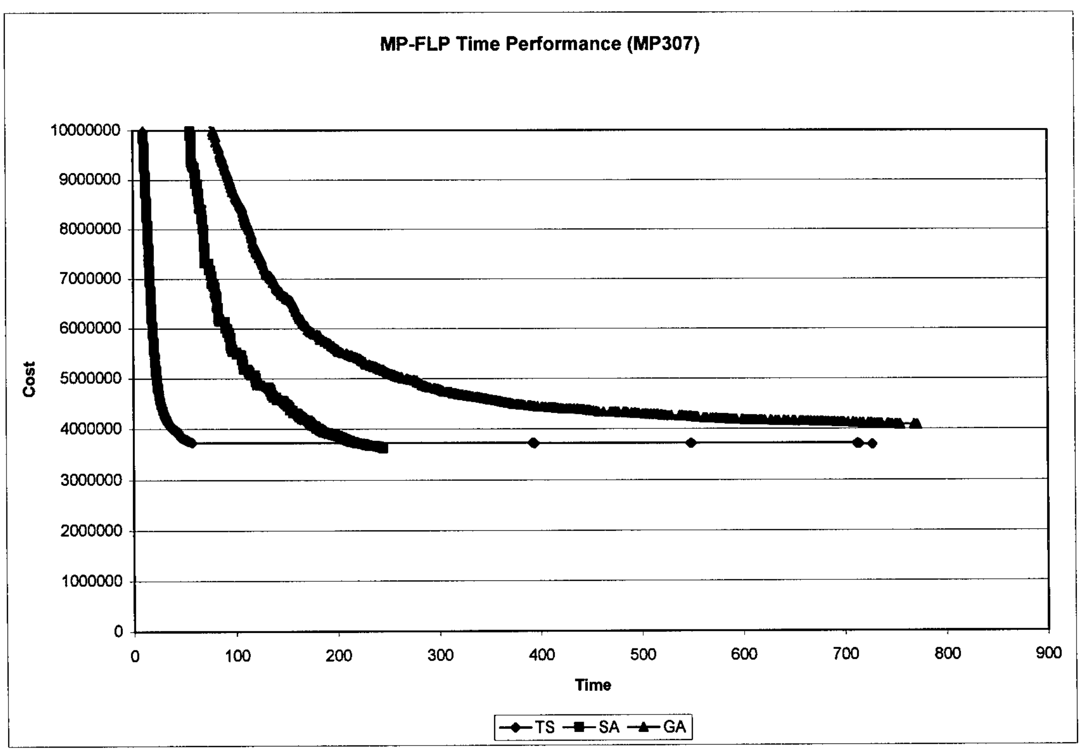

Figure 213. MP-FLP Time Performance Close-up (MP307)

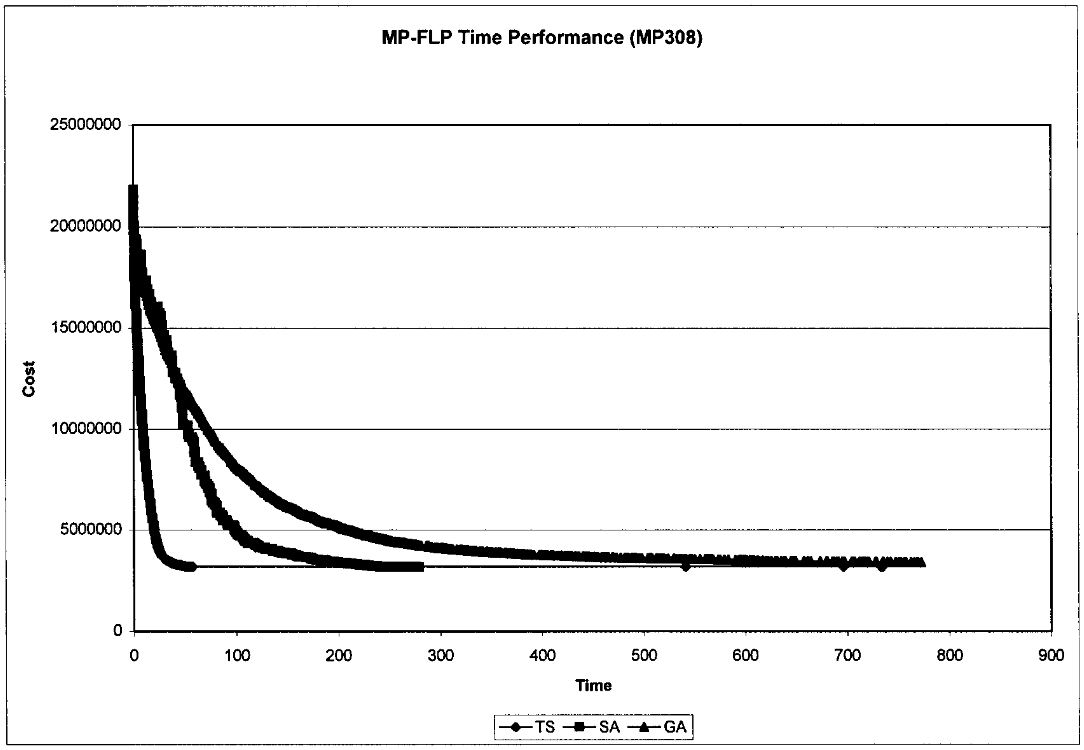

Figure 214. MP-FLP Time Performance (MP308)

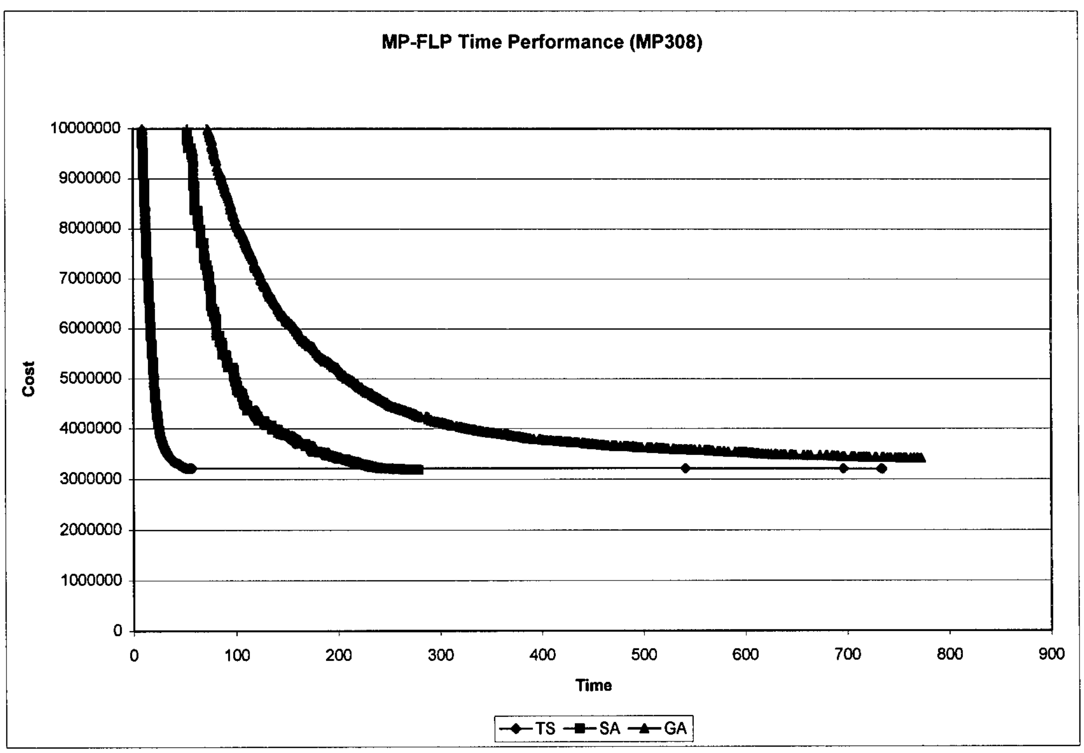

Figure 215. MP-FLP Time Performance Close-up (MP308)

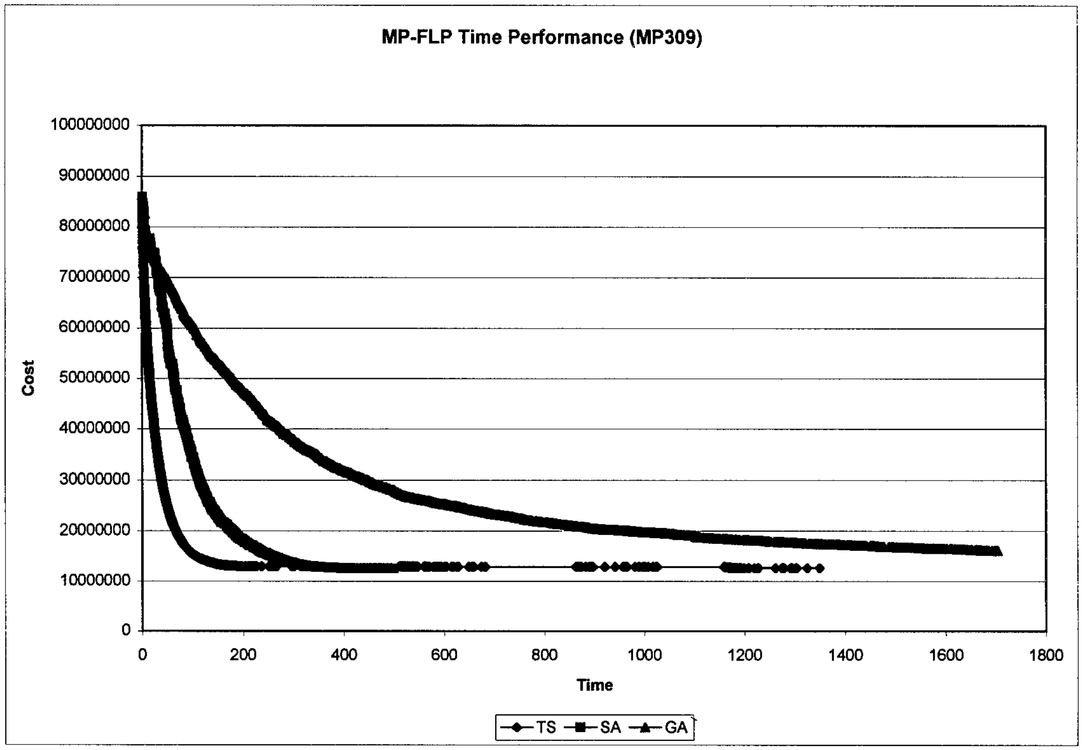

Figure 216. MP-FLP Time Performance (MP309)

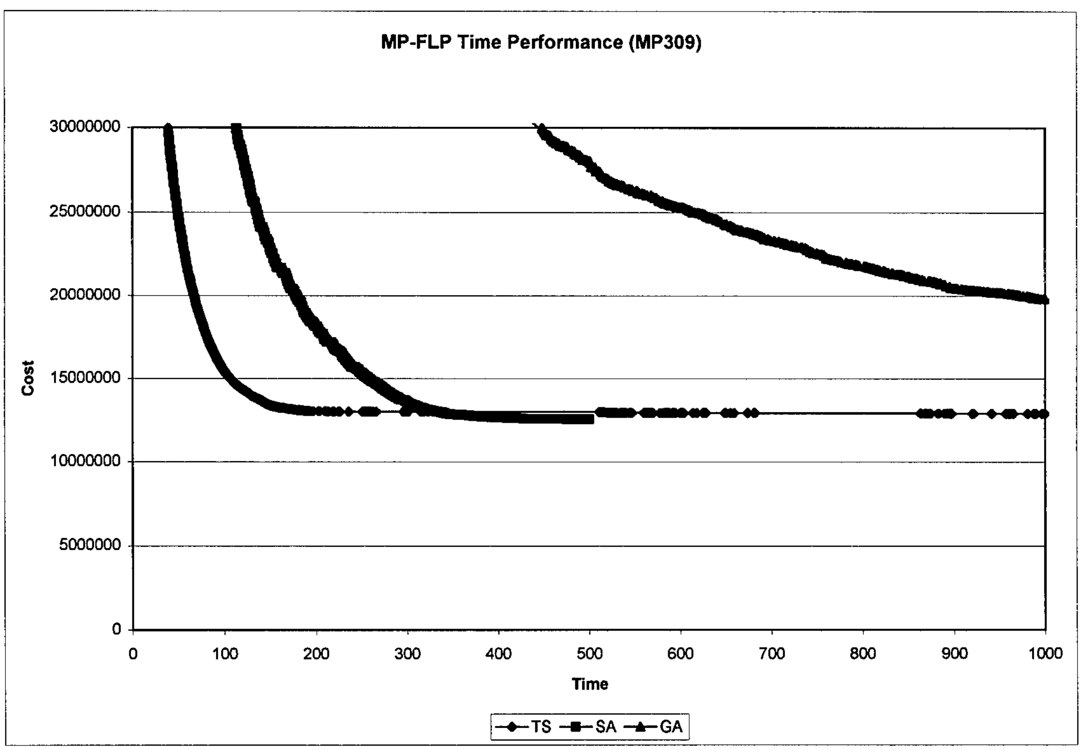

Figure 217. MP-FLP Time Performance Close-up (MP309)

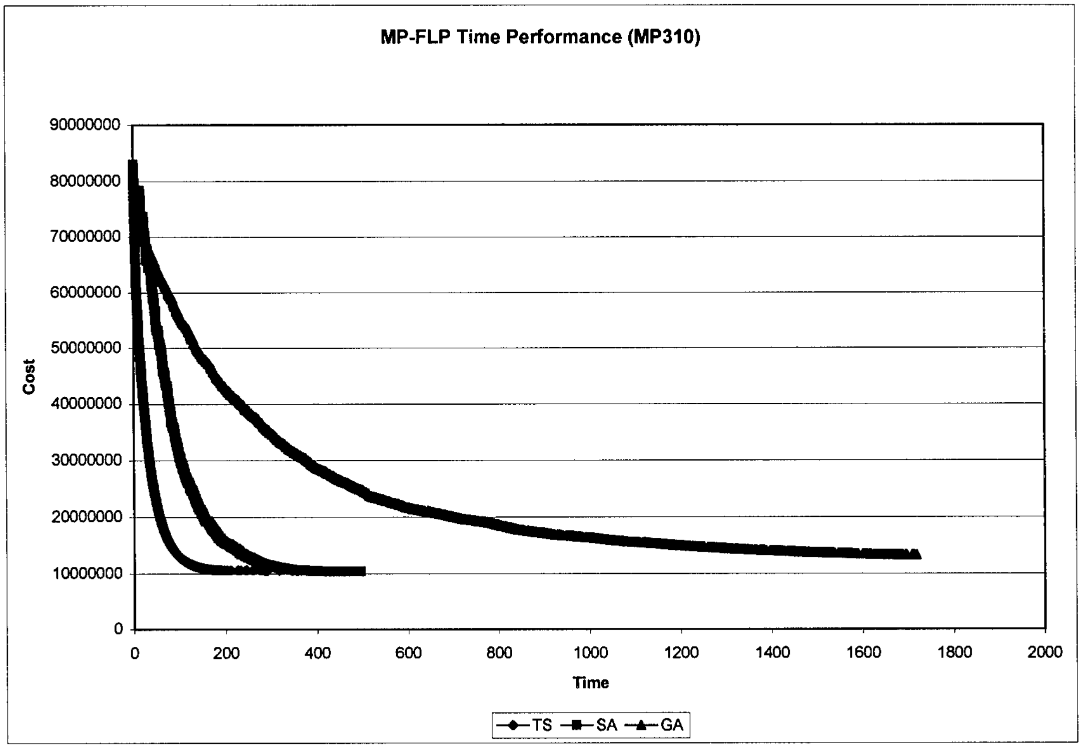

Figure 218. MP-FLP Time Performance (MP310)

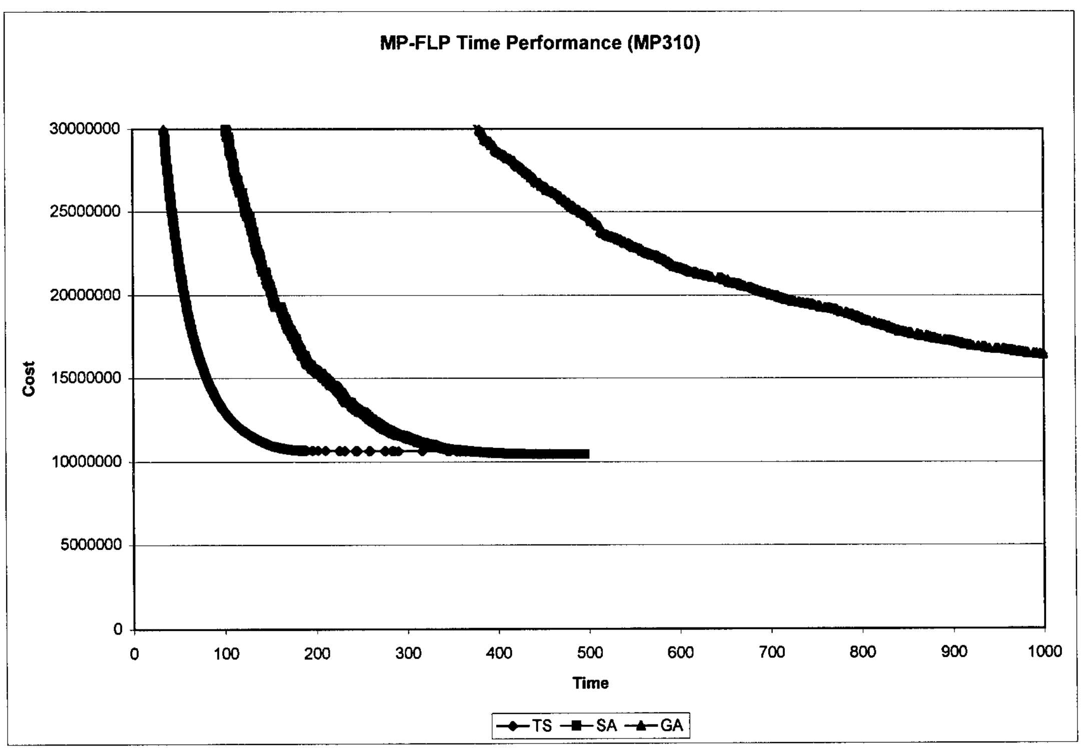

Figure 219. MP-FLP Time Performance Close-up (MP310)

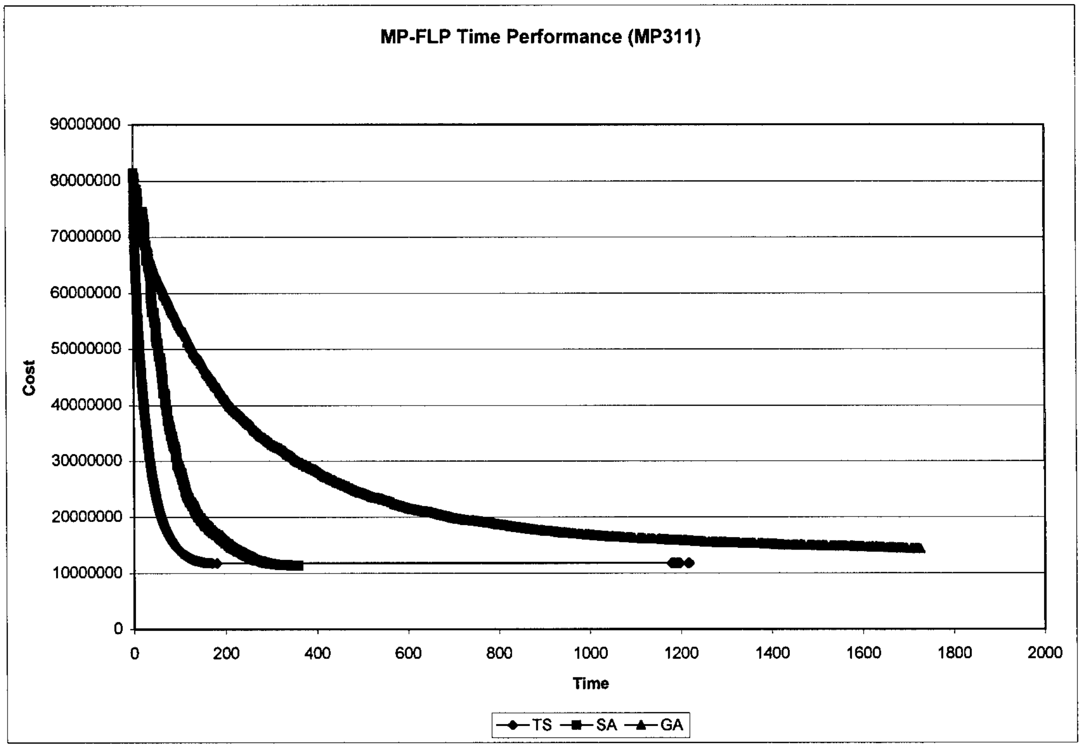

Figure 220. MP-FLP Time Performance (MP311)

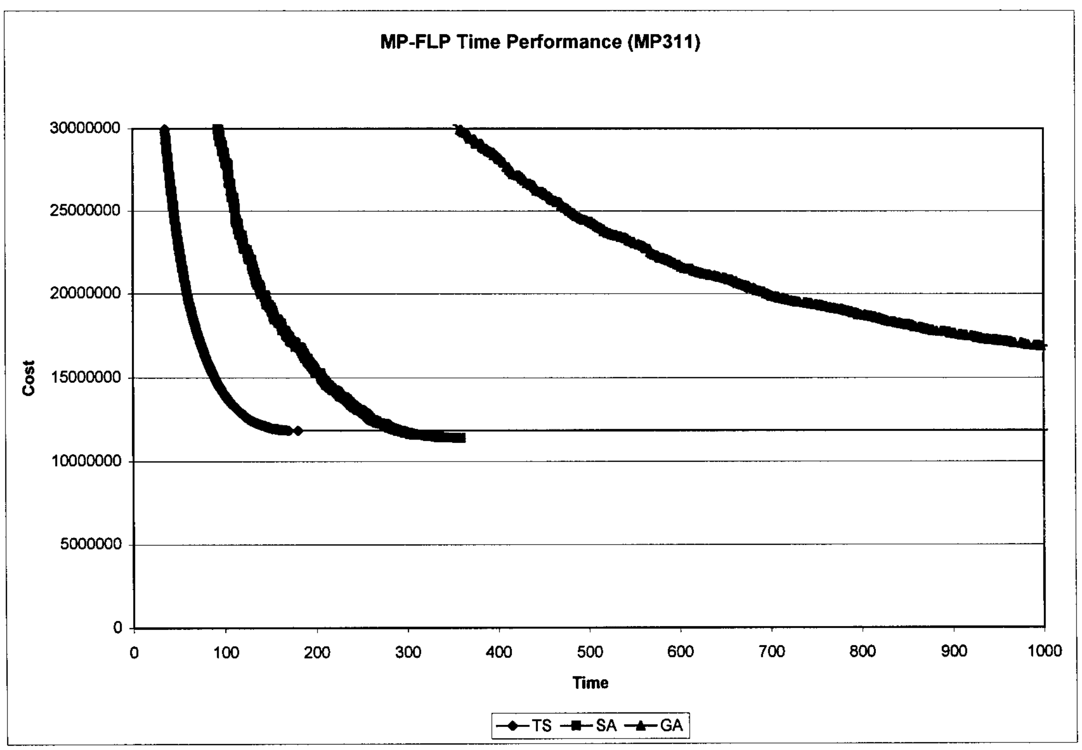

Figure 221. MP-FLP Time Performance Close-up (MP311)

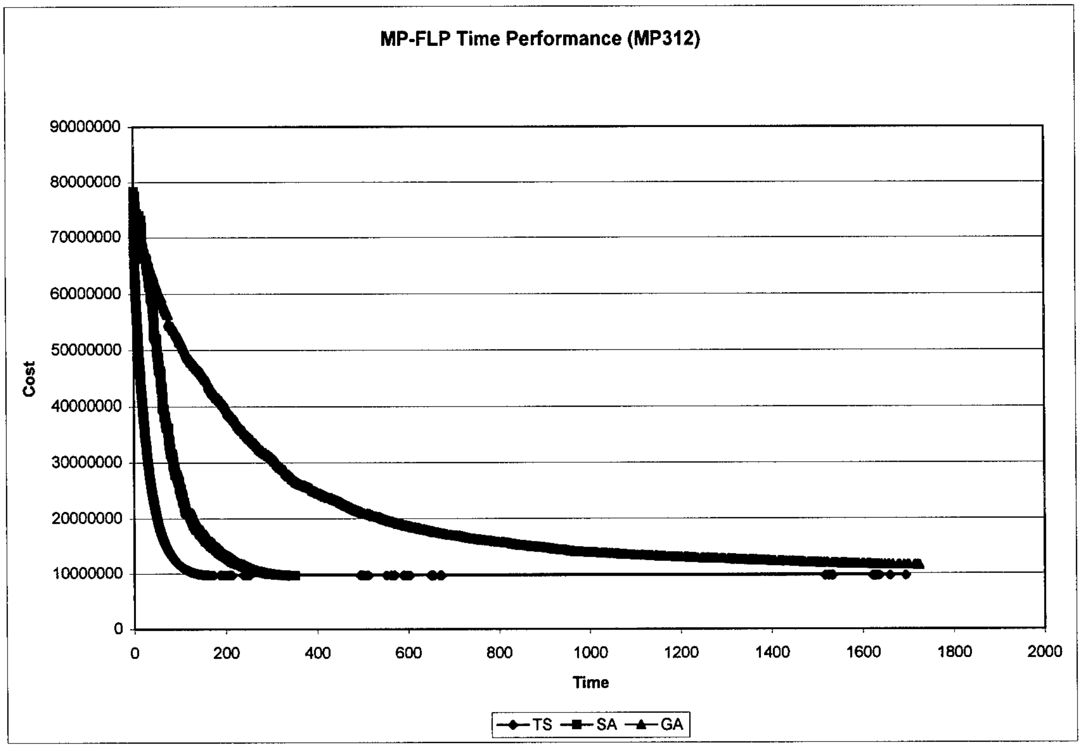

Figure 222. MP-FLP Time Performance (MP312)

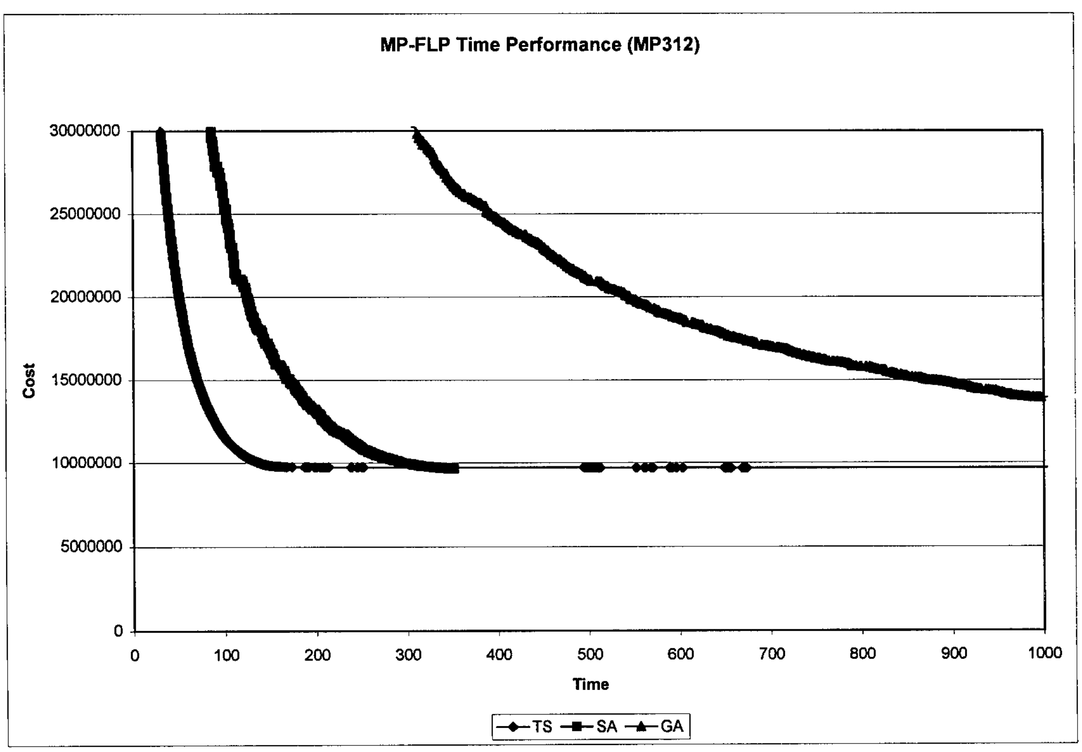

Figure 223. MP-FLP Time Performance Close-up (MP312)

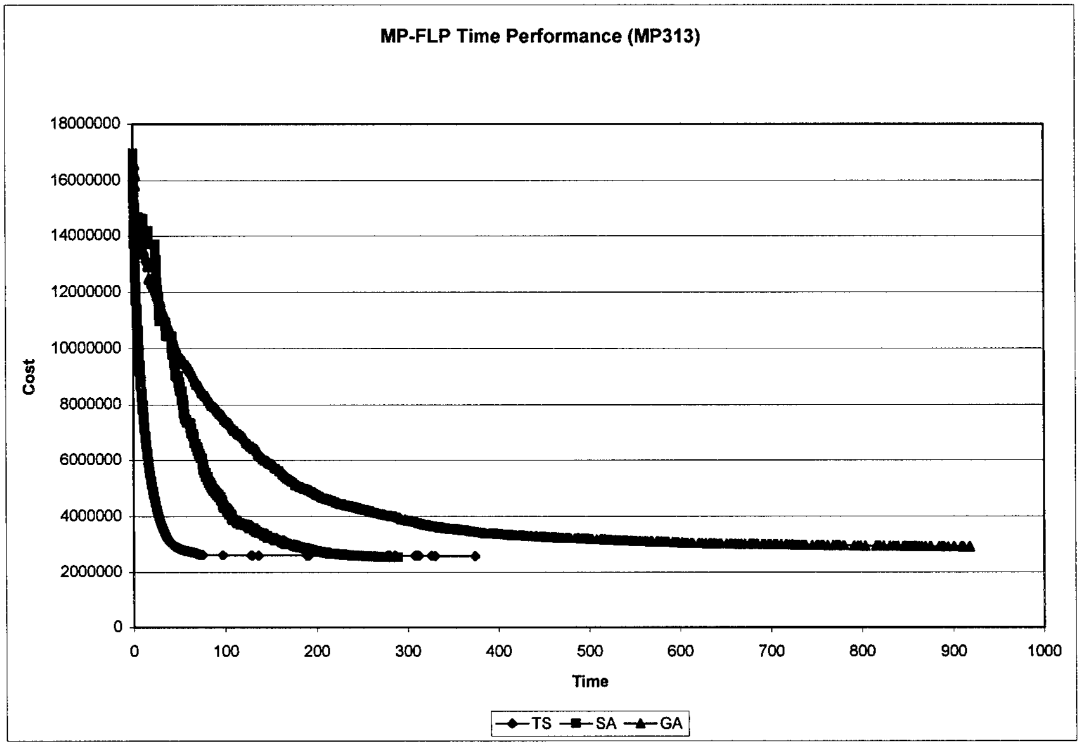

Figure 224. MP-FLP Time Performance (MP313)

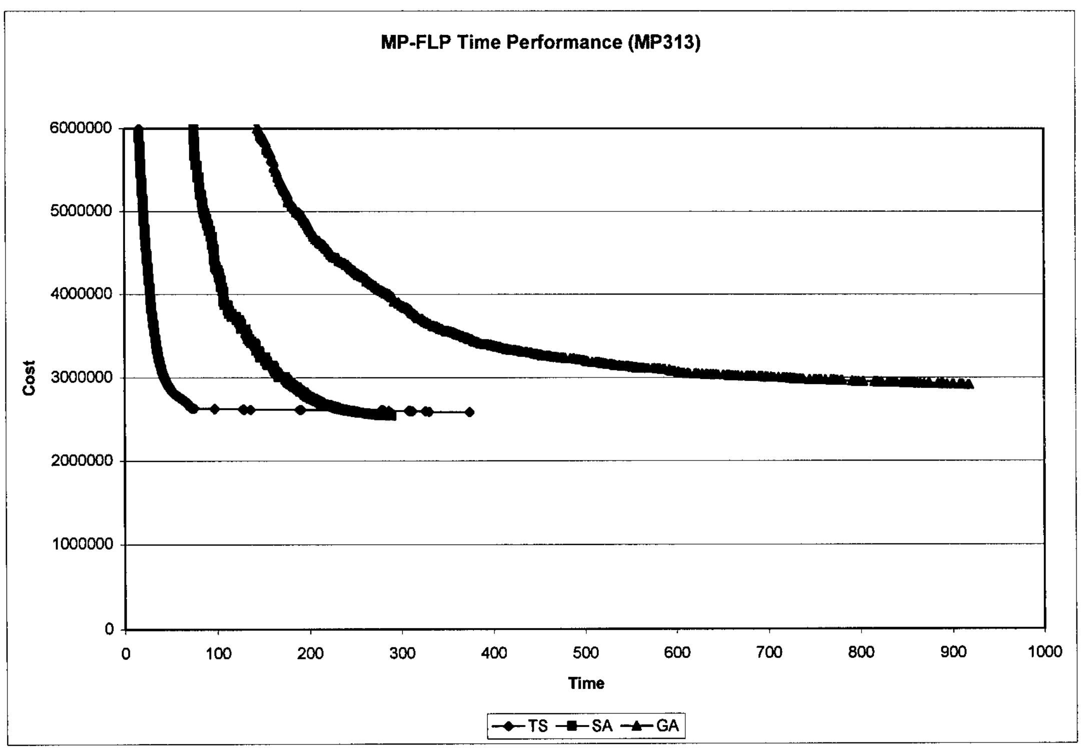

Figure 225. MP-FLP Time Performance Close-up (MP313)

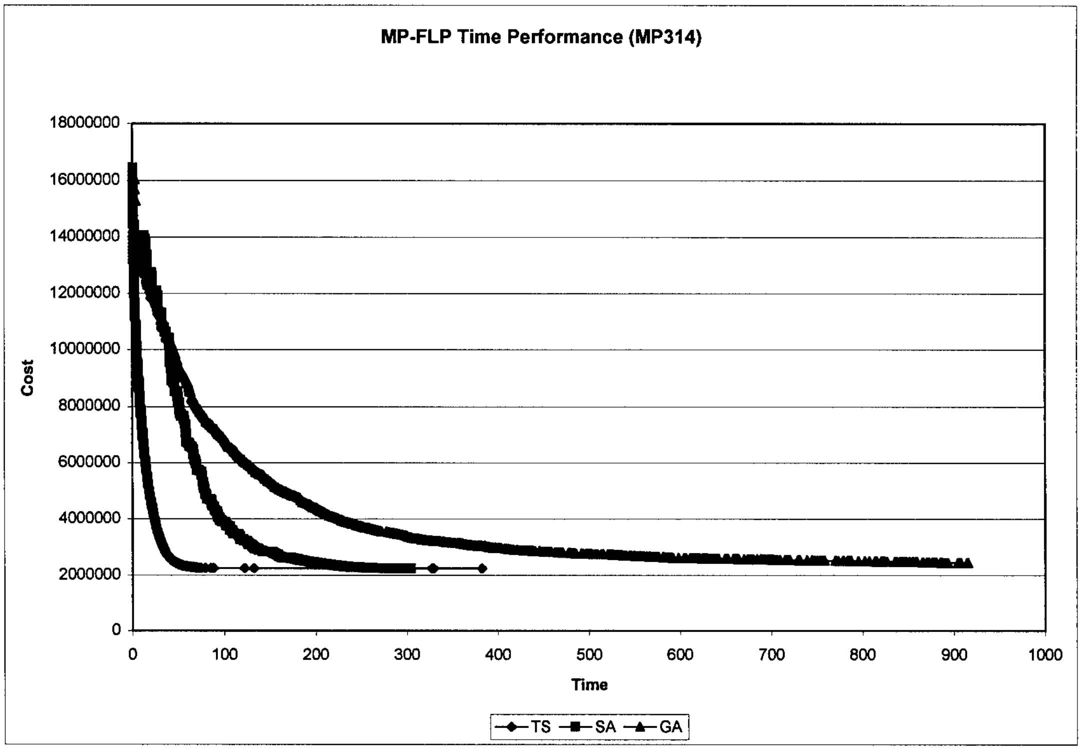

Figure 226. MP-FLP Time Performance (MP314)

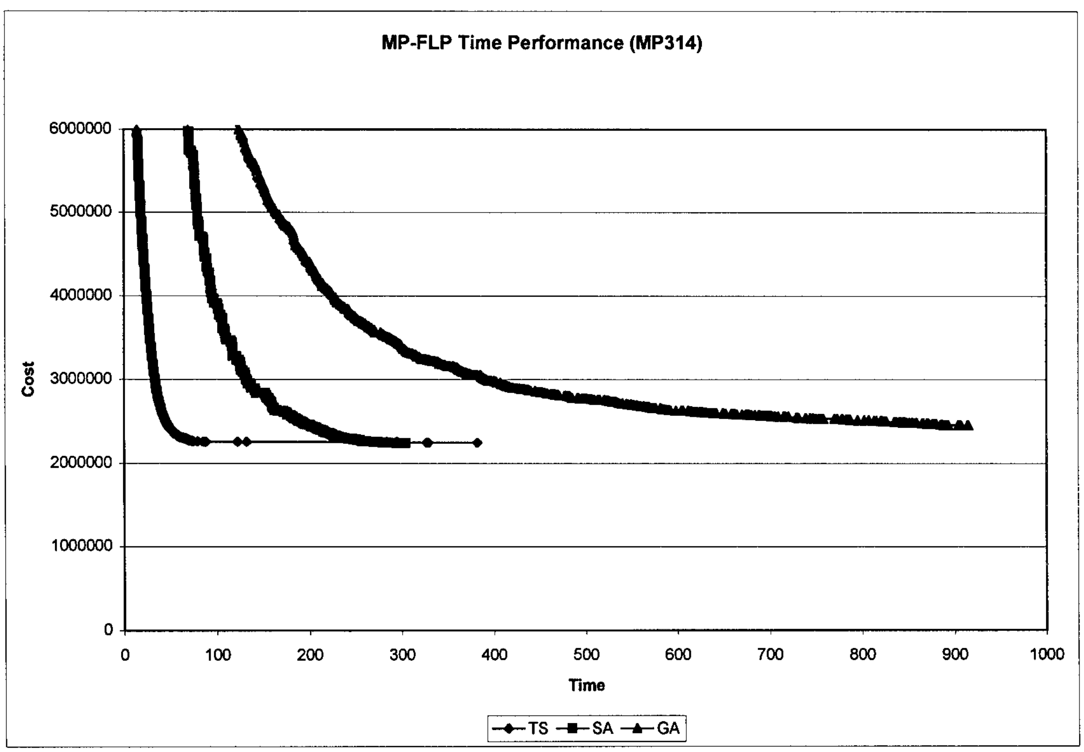

Figure 227. MP-FLP Time Performance Close-up (MP314)

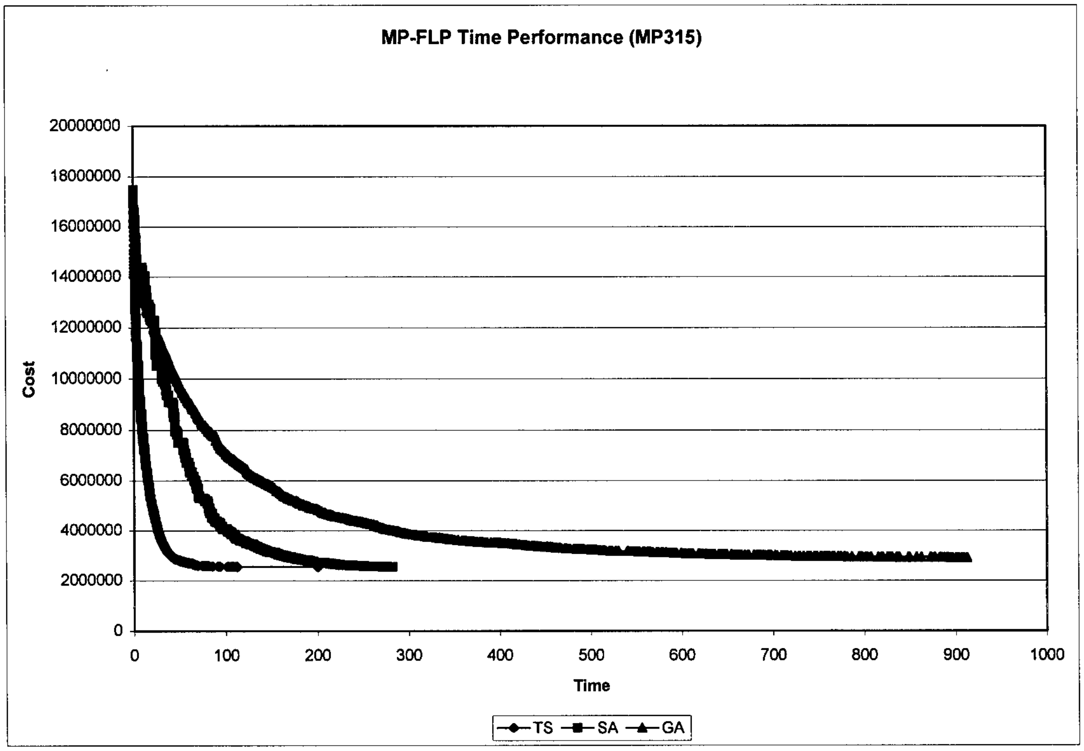

Figure 228. MP-FLP Time Performance (MP315)

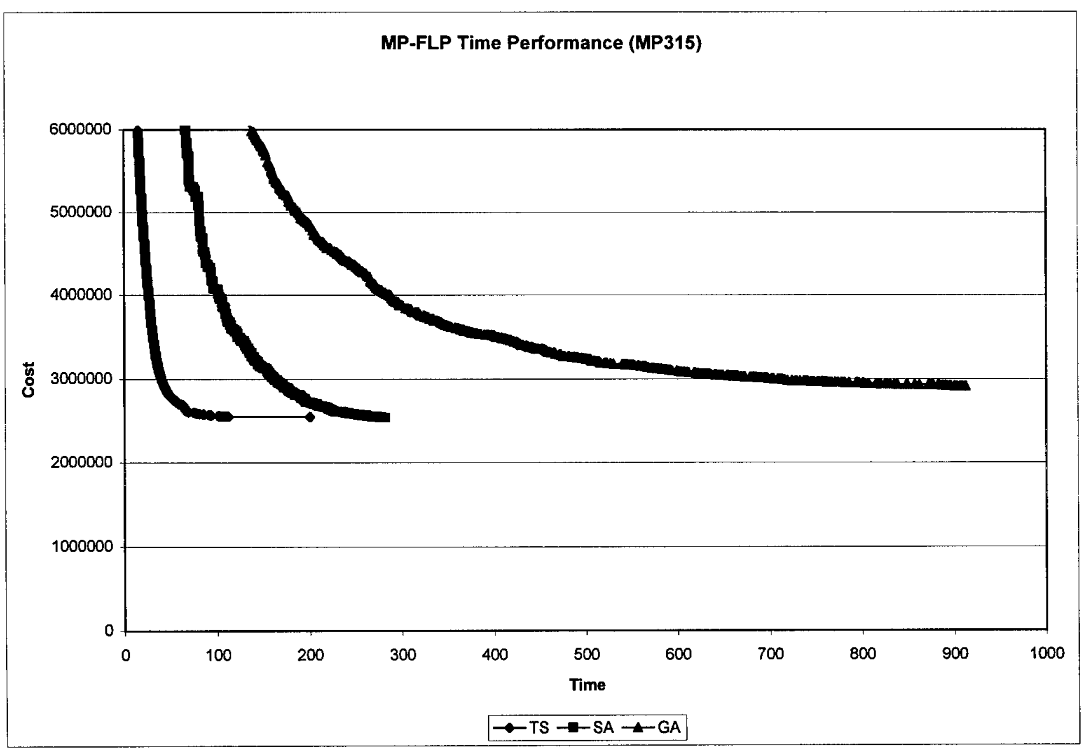

Figure 229. MP-FLP Time Performance Close-up (MP315)

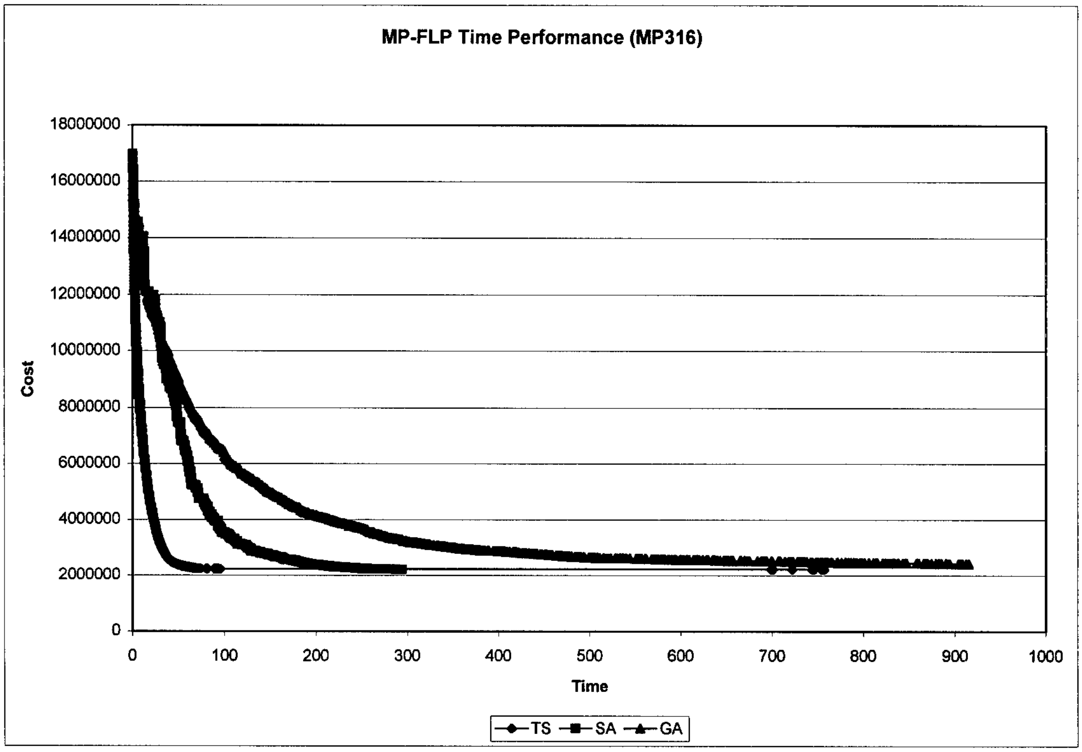

Figure 230. MP-FLP Time Performance (MP316)

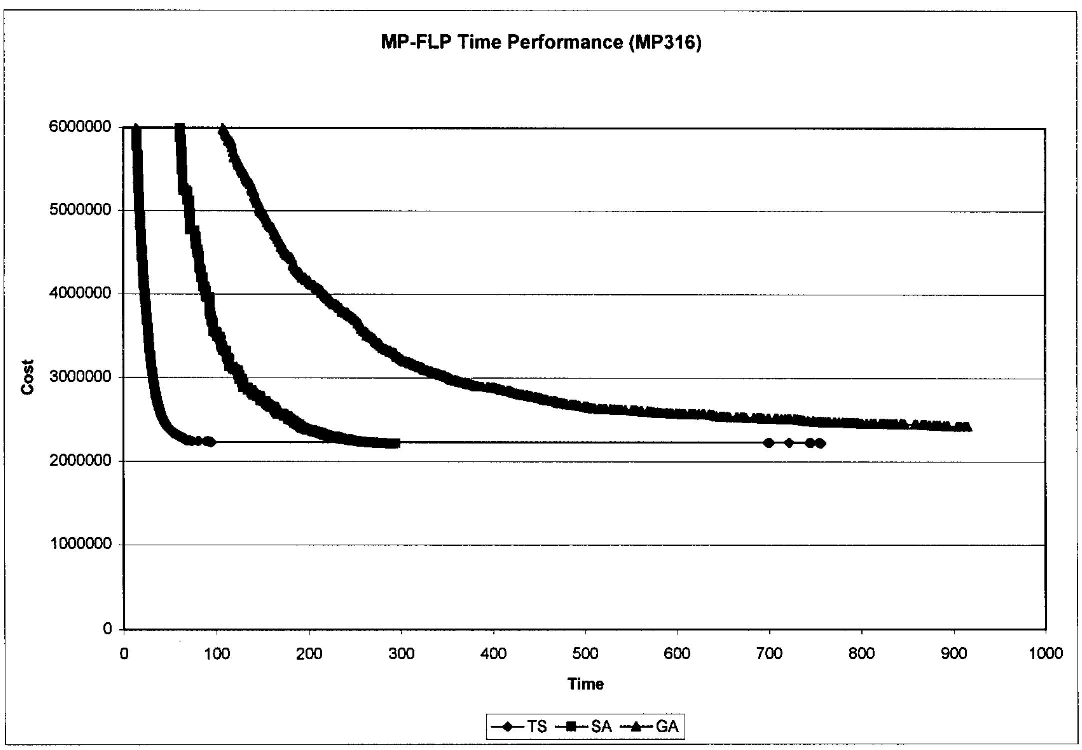

Figure 231. MP-FLP Time Performance Close-up (MP316)

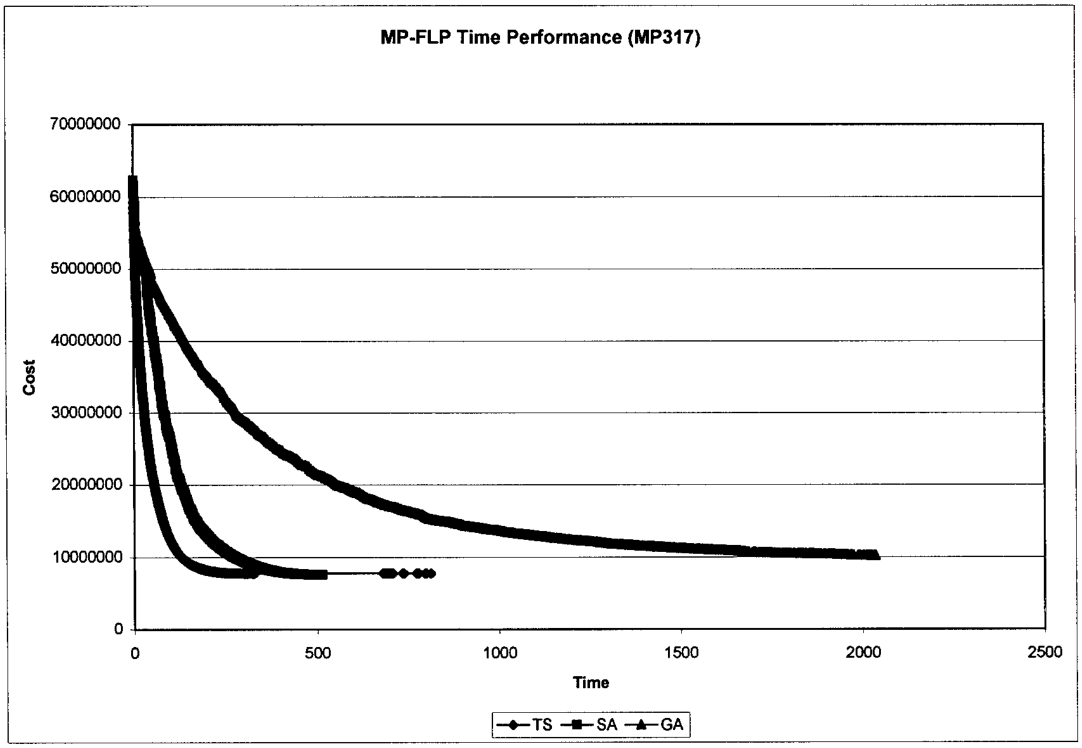

Figure 232. MP-FLP Time Performance (MP317)

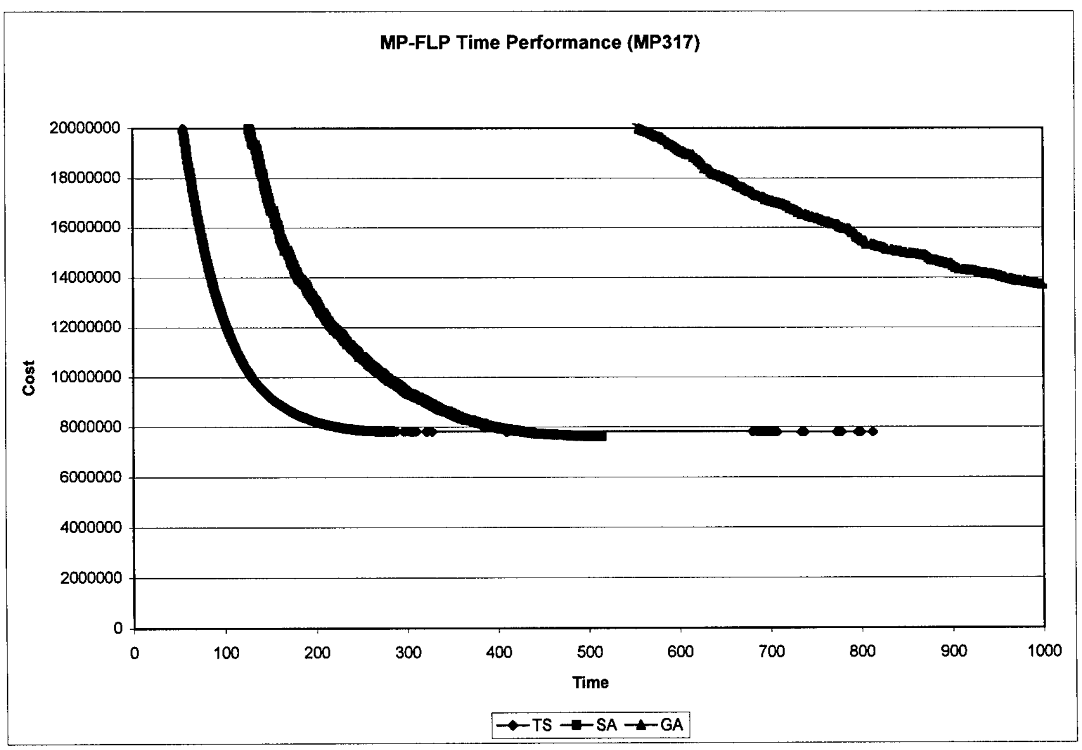

Figure 233. MP-FLP Time Performance Close-up (MP317)

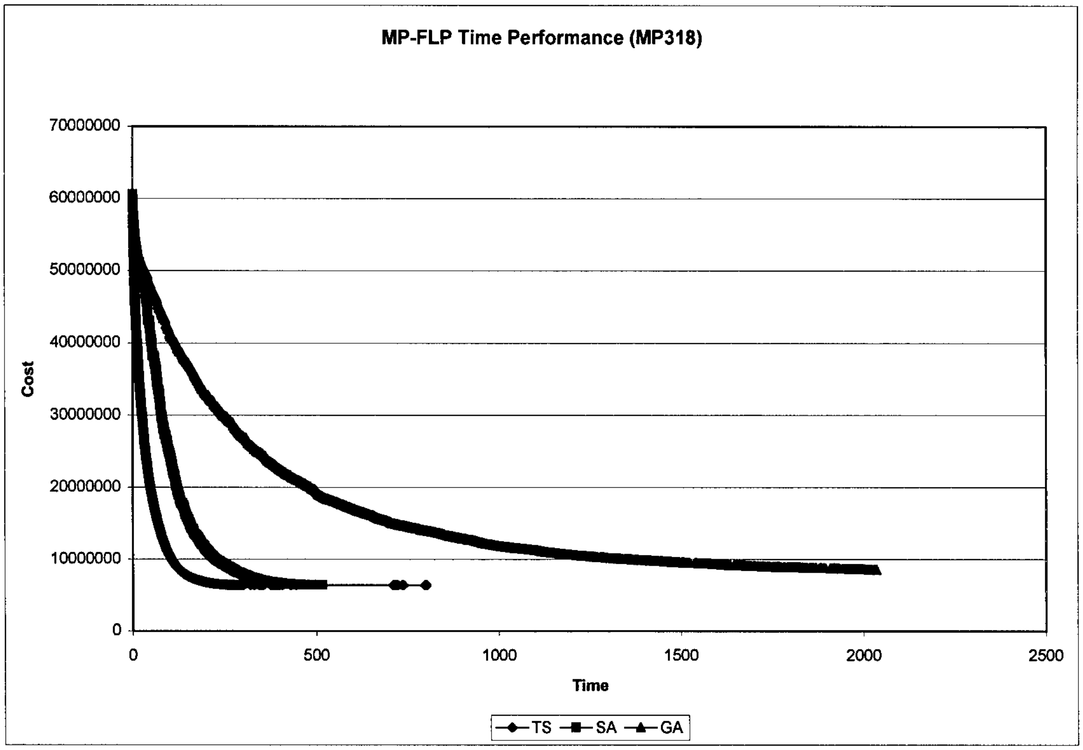

Figure 234. MP-FLP Time Performance (MP3018)

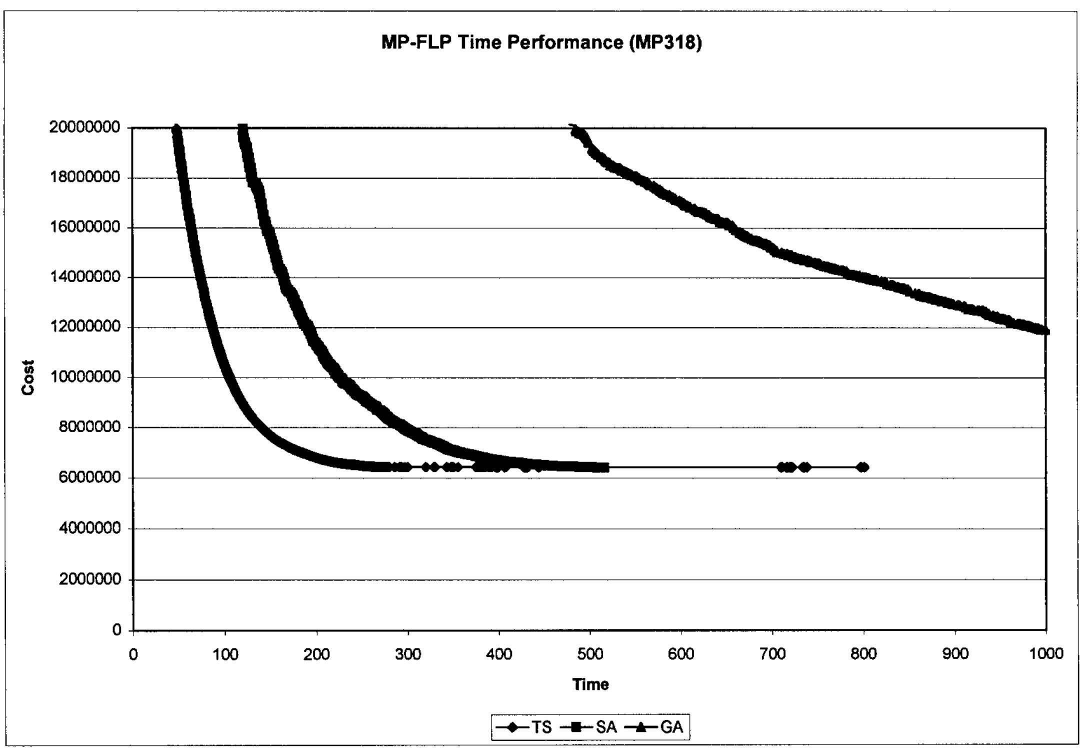

Figure 235. MP-FLP Time Performance Close-up (MP318)

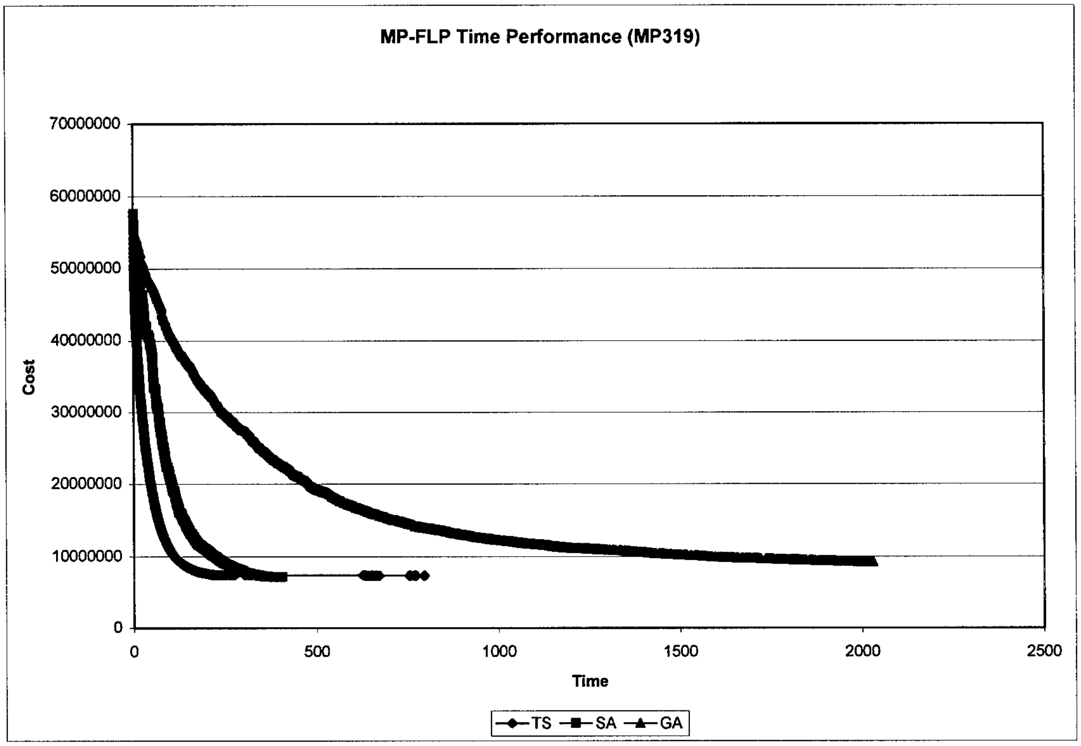

Figure 236. MP-FLP Time Performance (MP329)

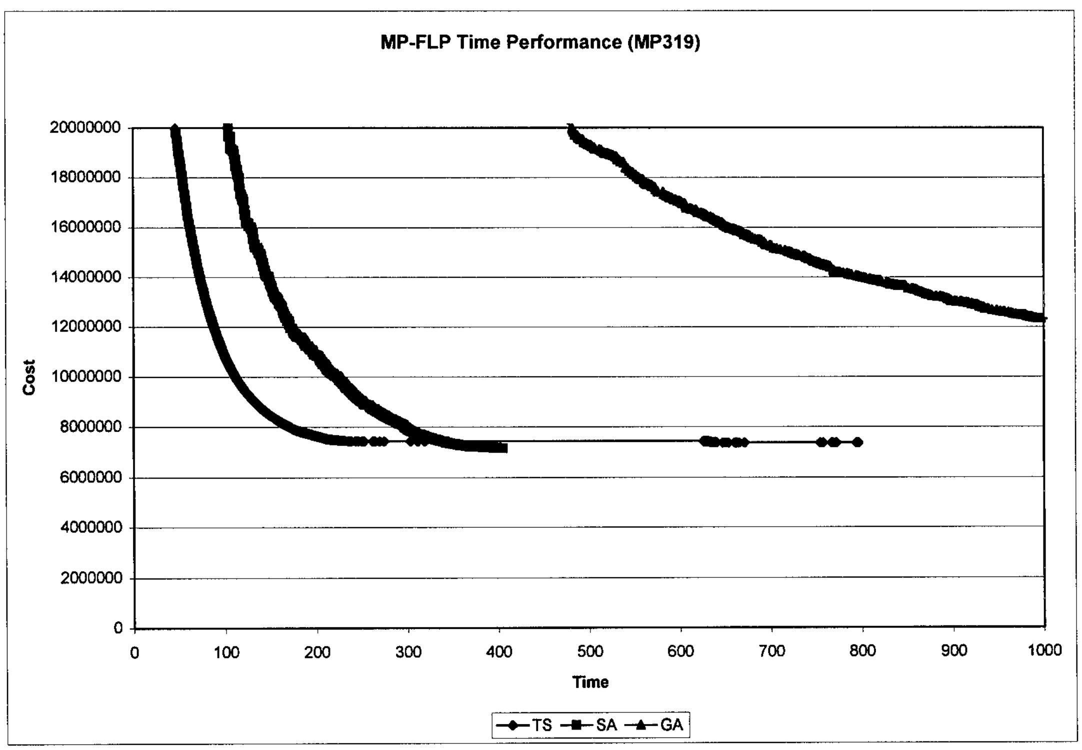

Figure 237. MP-FLP Time Performance Close-up (MP319)

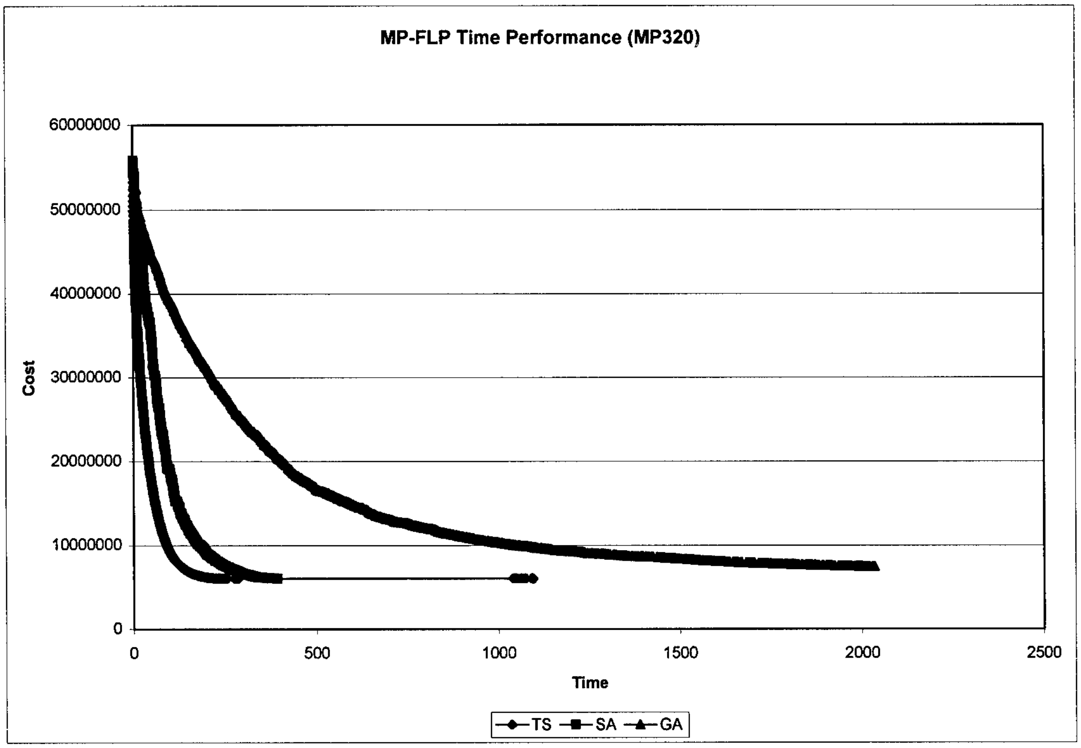

Figure 238. MP-FLP Time Performance (MP320)

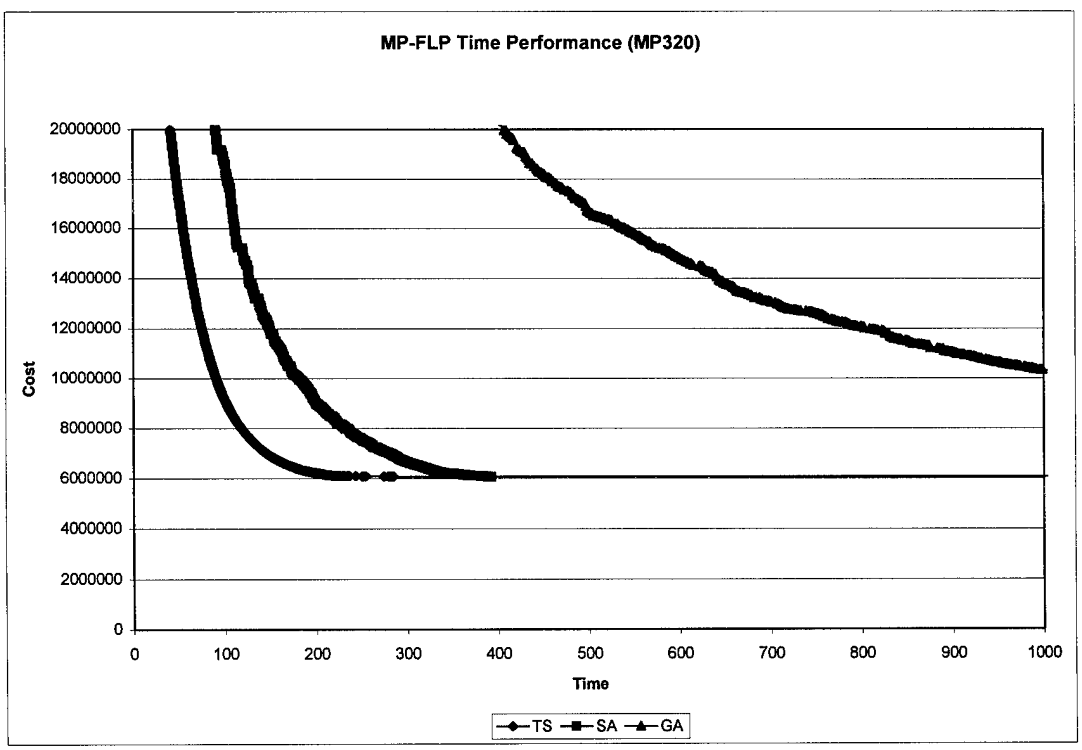

Figure 239. MP-FLP Time Performance Close-up (MP320)

APPENDIX H

MP-FLP SOLUTIONS PERFORMANCE CHARTS

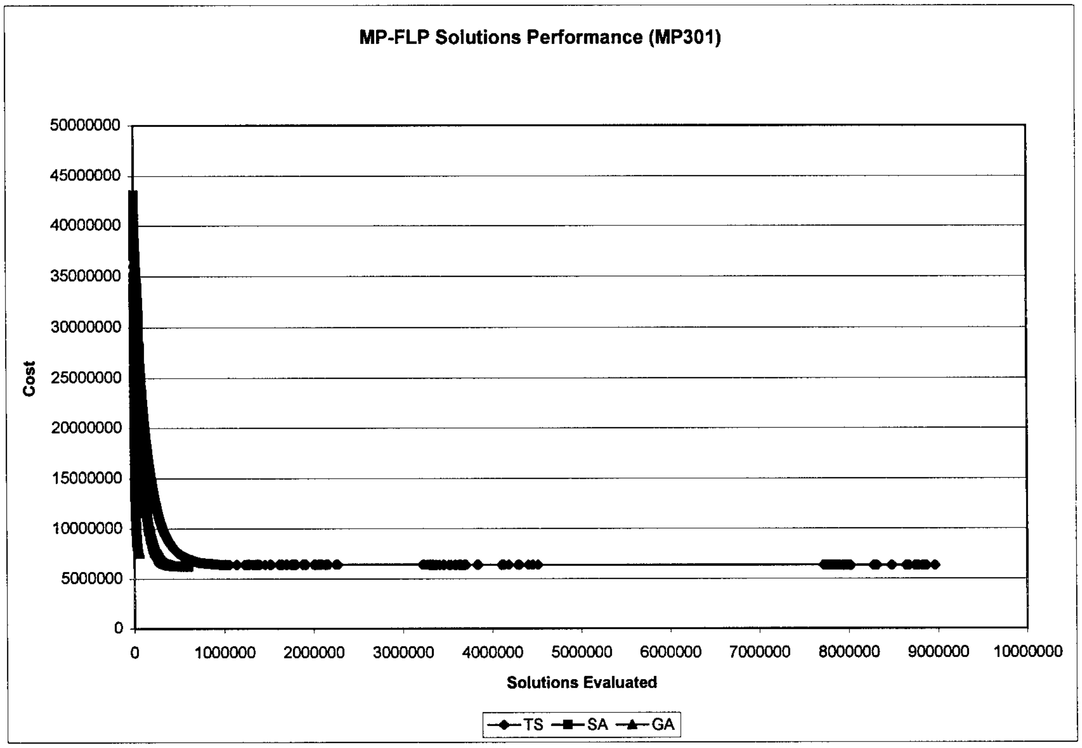

Figure 240. MP-FLP Solutions Performance (MP301)

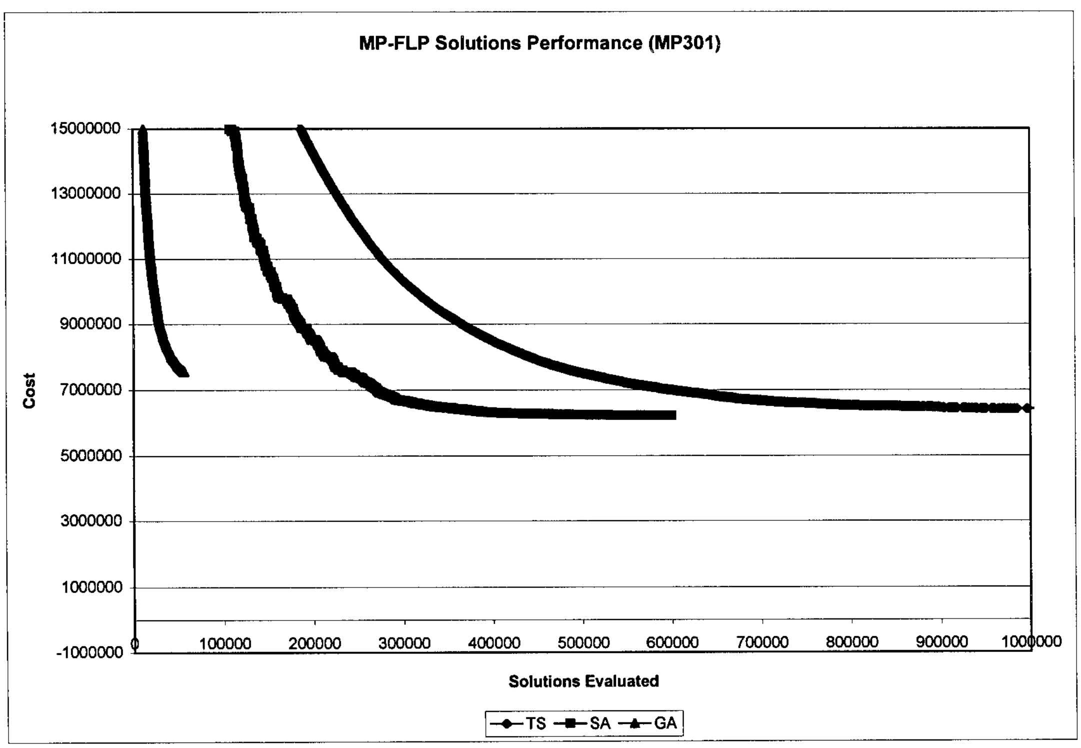

Figure 241. MP-FLP Solutions Performance Close-up (MP301)

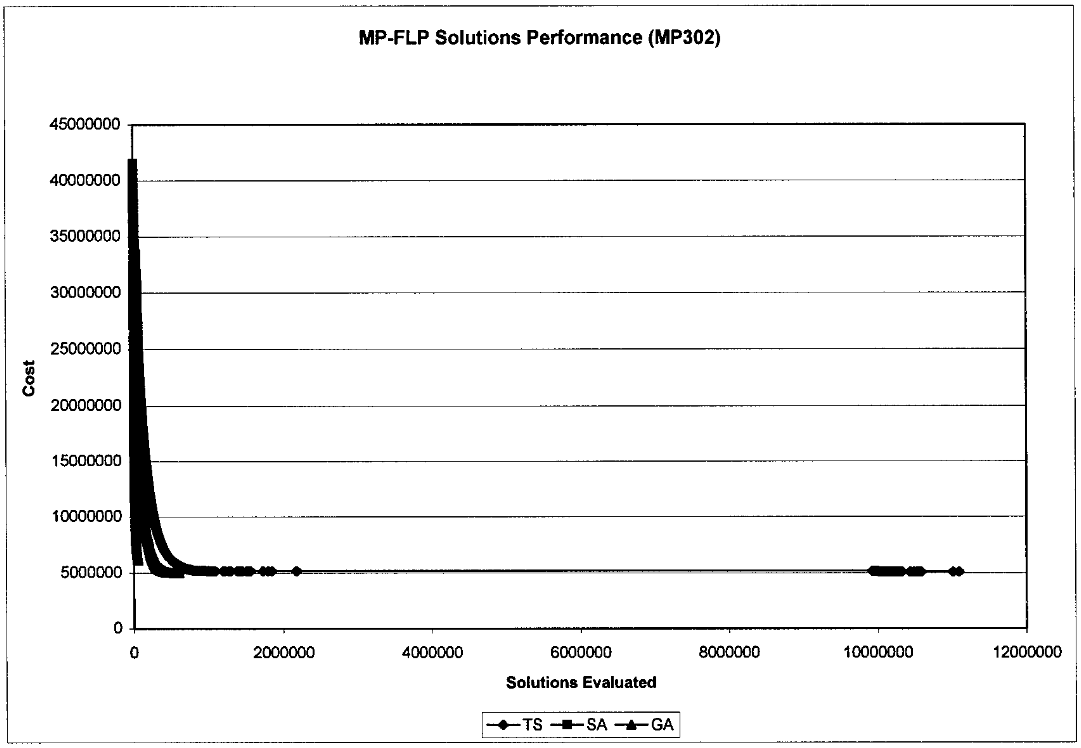

Figure 242. MP-FLP Solutions Performance (MP302)

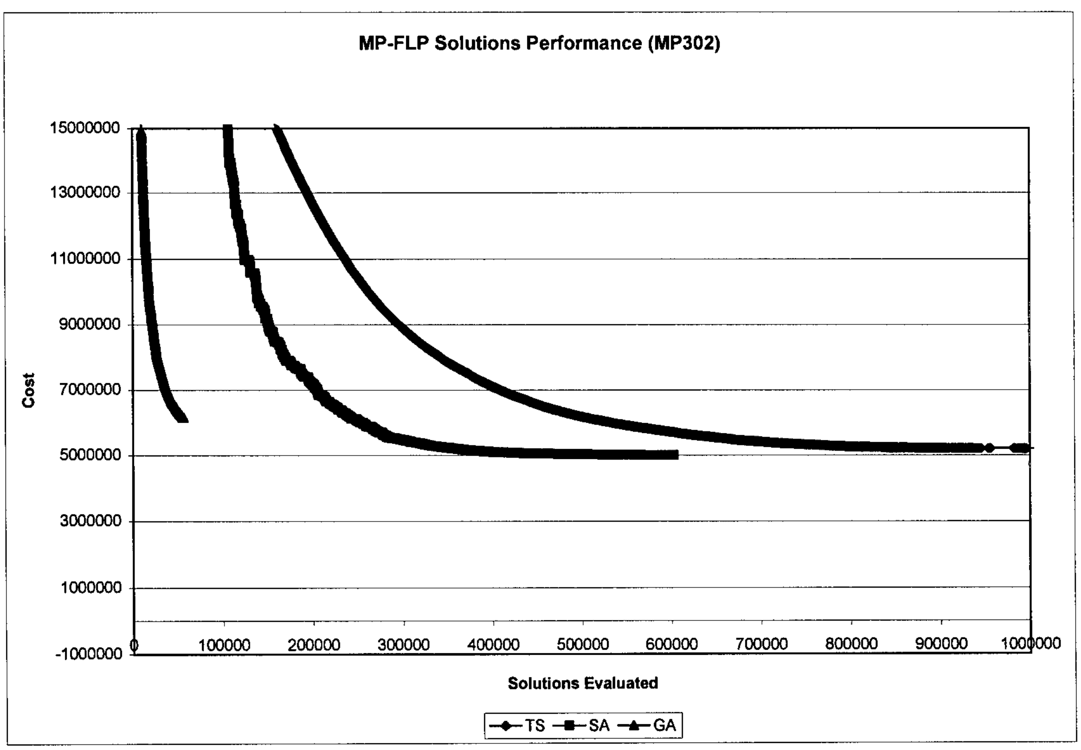

Figure 243. MP-FLP Solutions Performance Close-up (MP302)

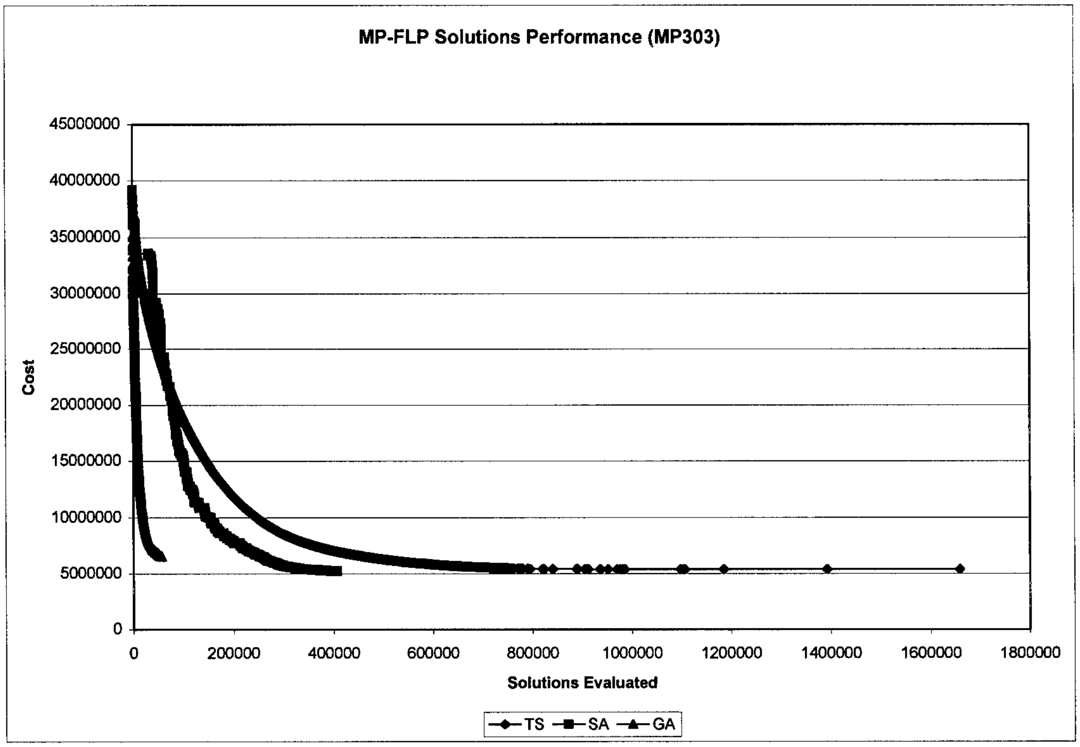

Figure 244. MP-FLP Solutions Performance (MP303)

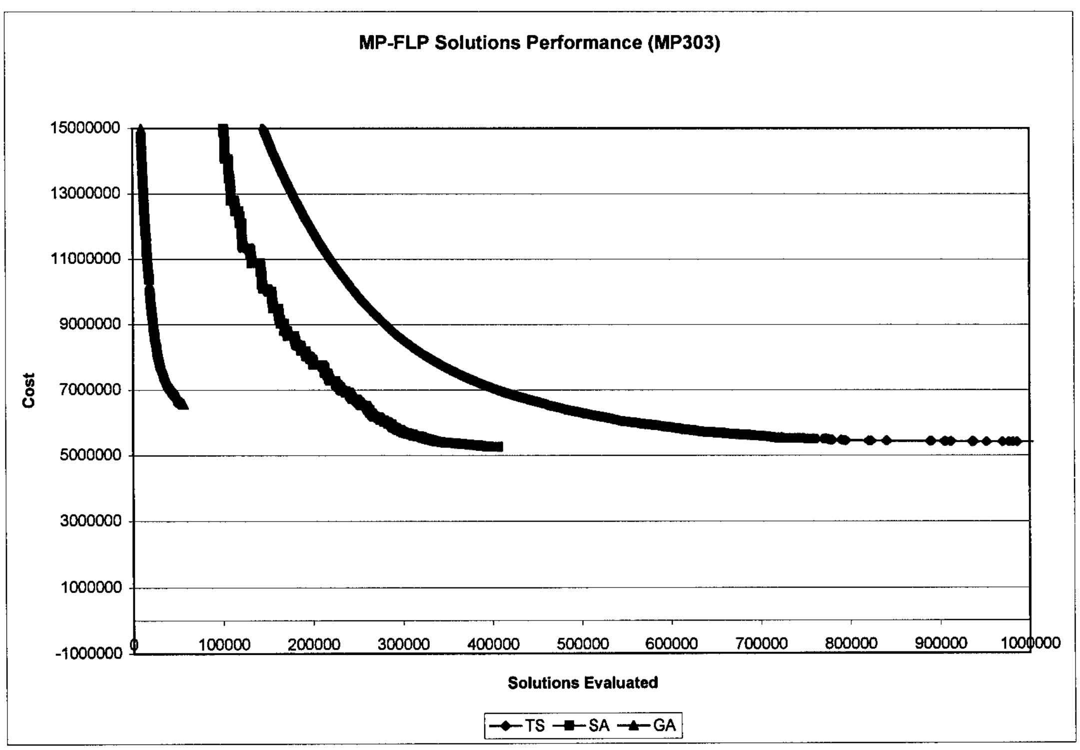

Figure 245. MP-FLP Solutions Performance Close-up (MP303)

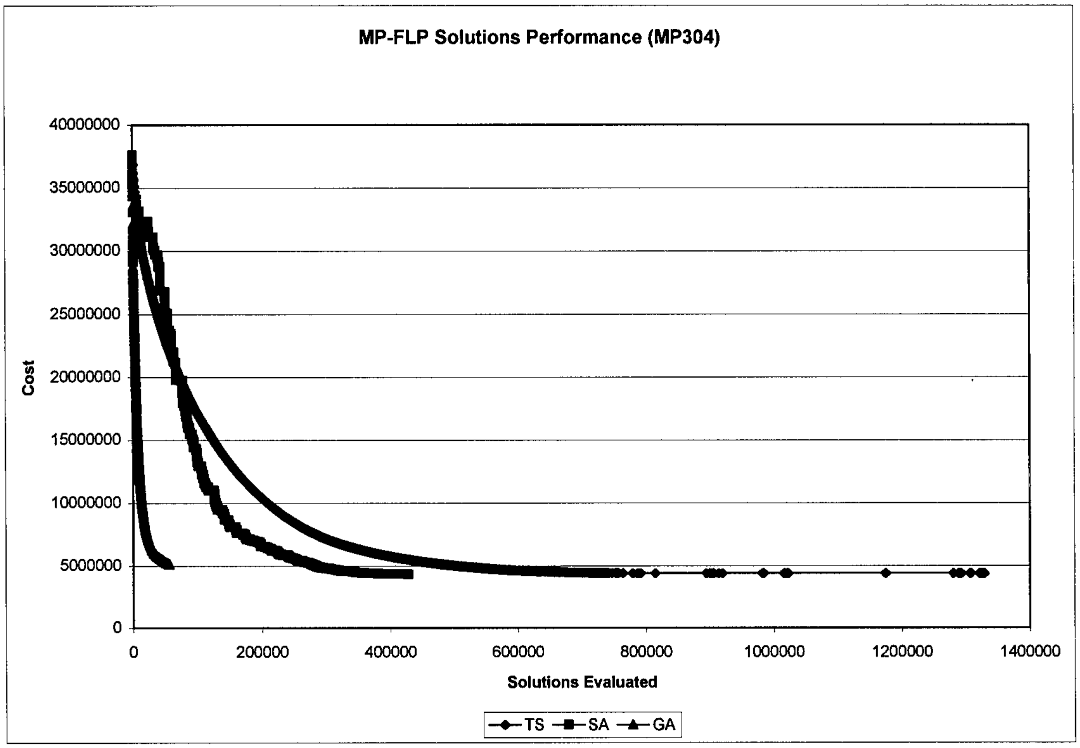

Figure 246. MP-FLP Solutions Performance (MP304)

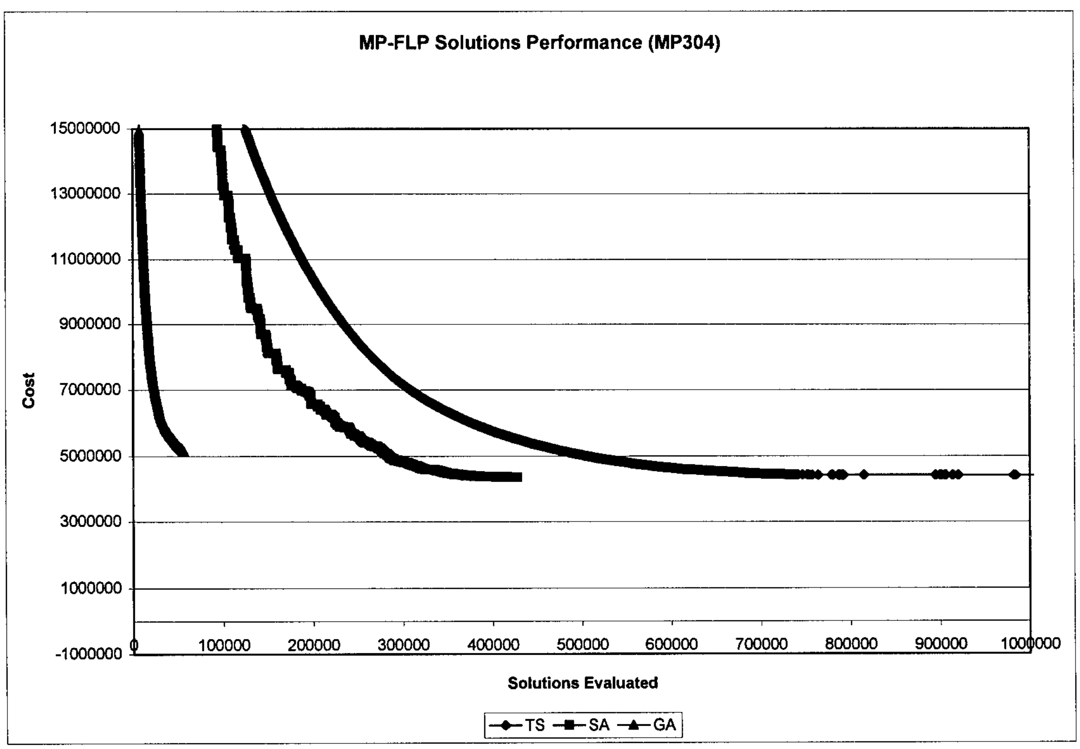

Figure 247. MP-FLP Solutions Performance Close-up (MP304)

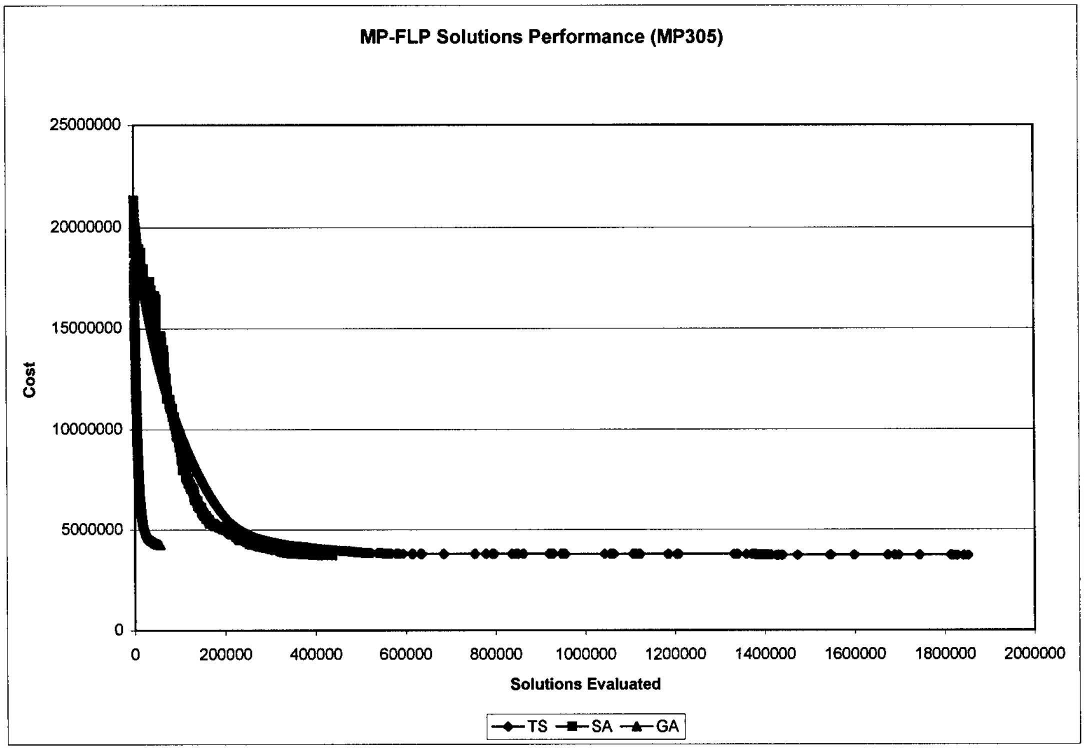

Figure 248. MP-FLP Solutions Performance (MP305)

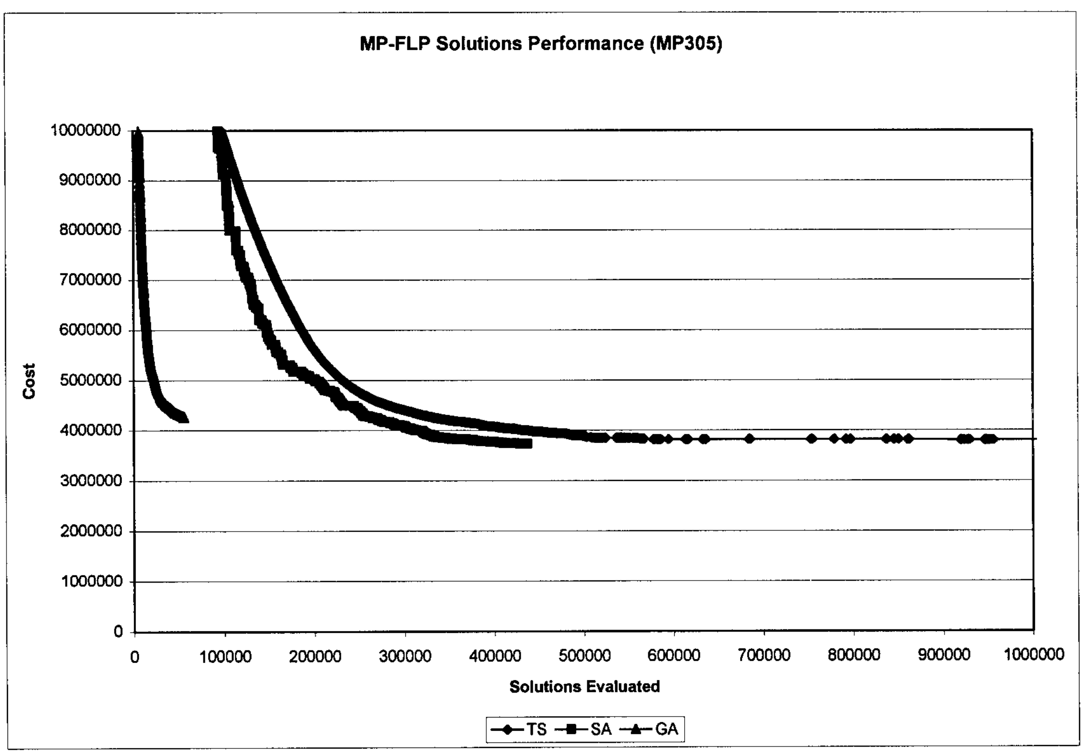

Figure 249. MP-FLP Solutions Performance Close-up (MP305)

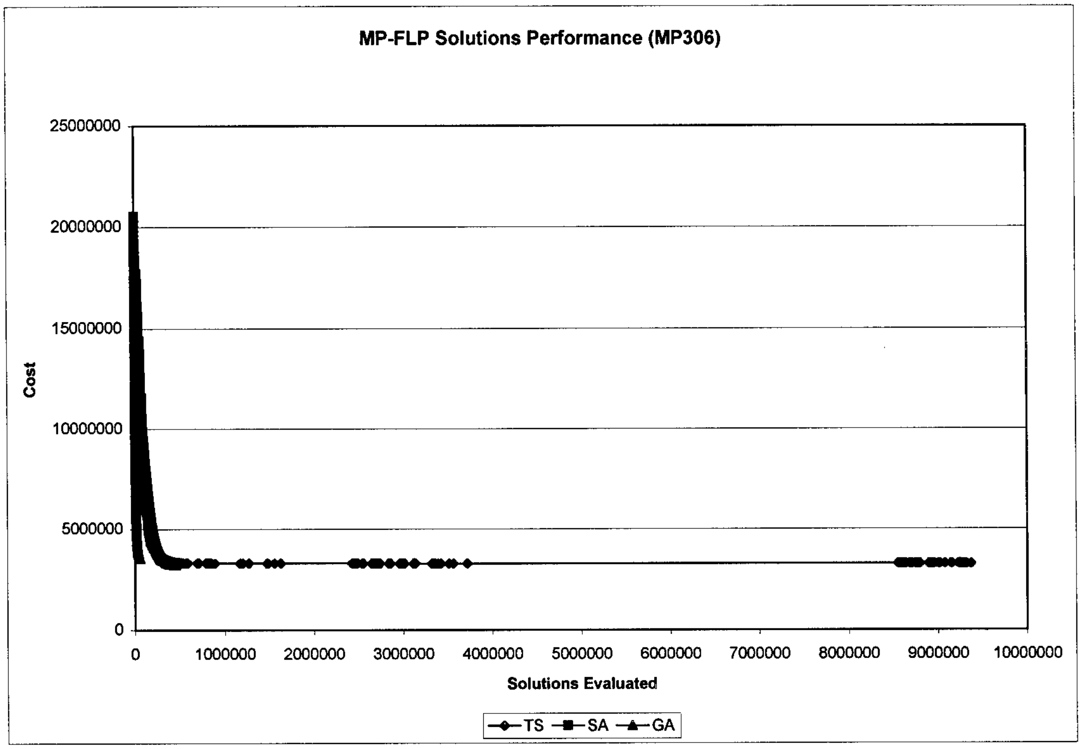

Figure 250. MP-FLP Solutions Performance (MP306)

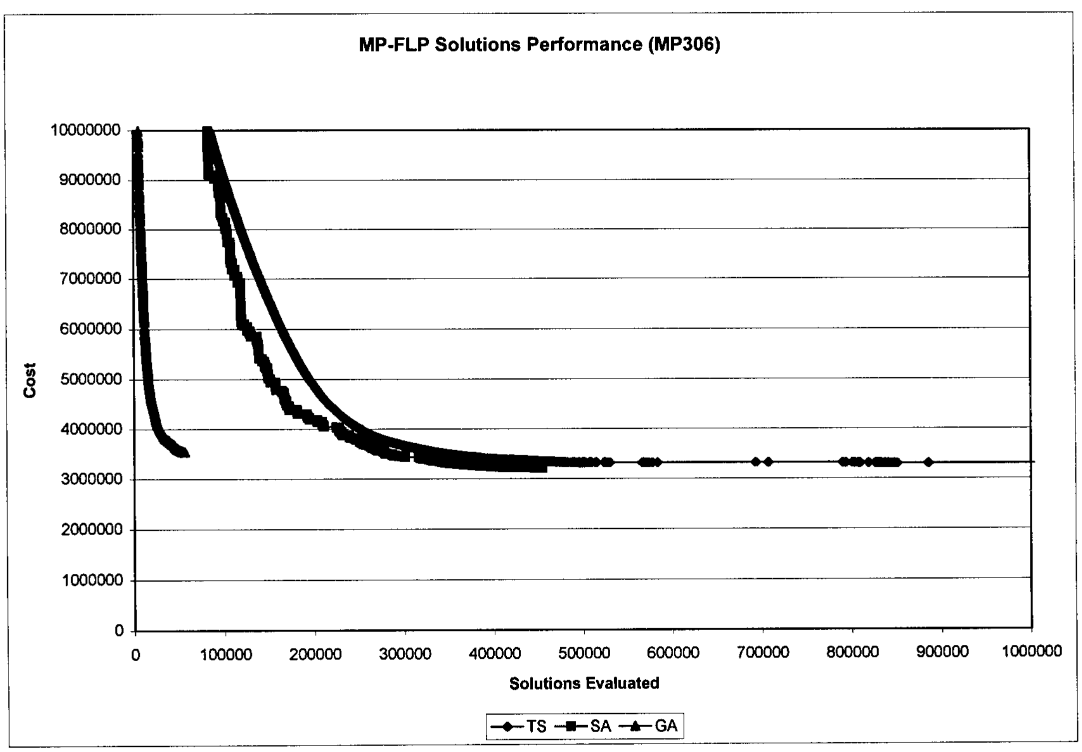

Figure 251. MP-FLP Solutions Performance Close-up (MP306)

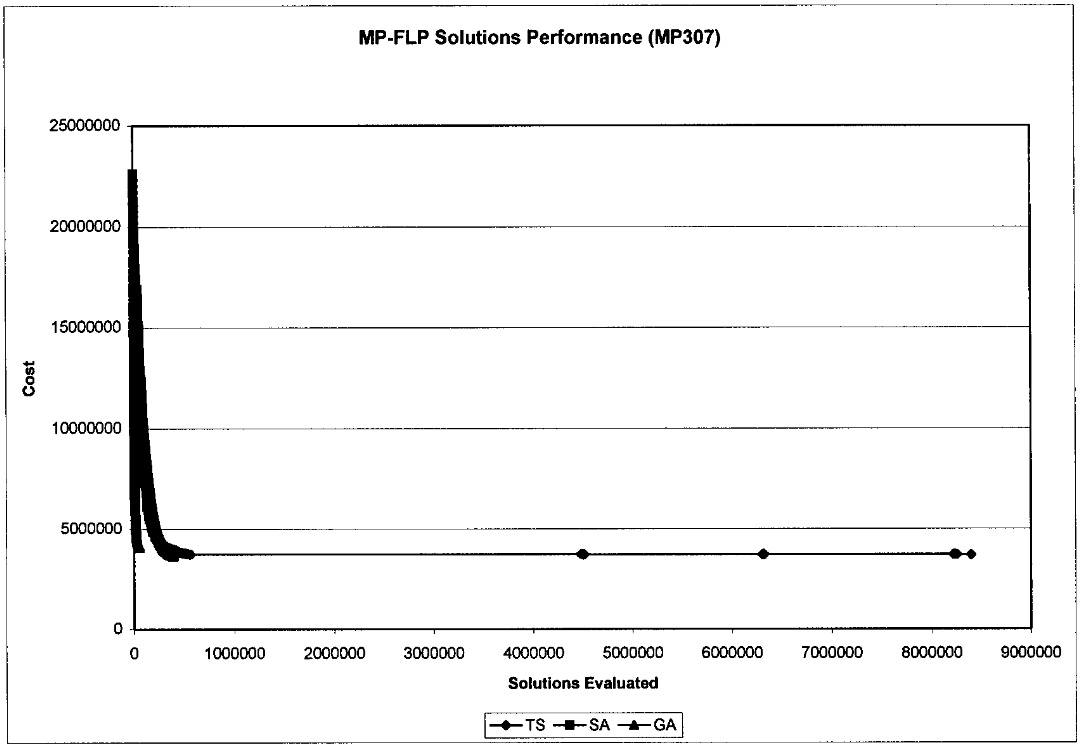

Figure 252. MP-FLP Solutions Performance (MP307)

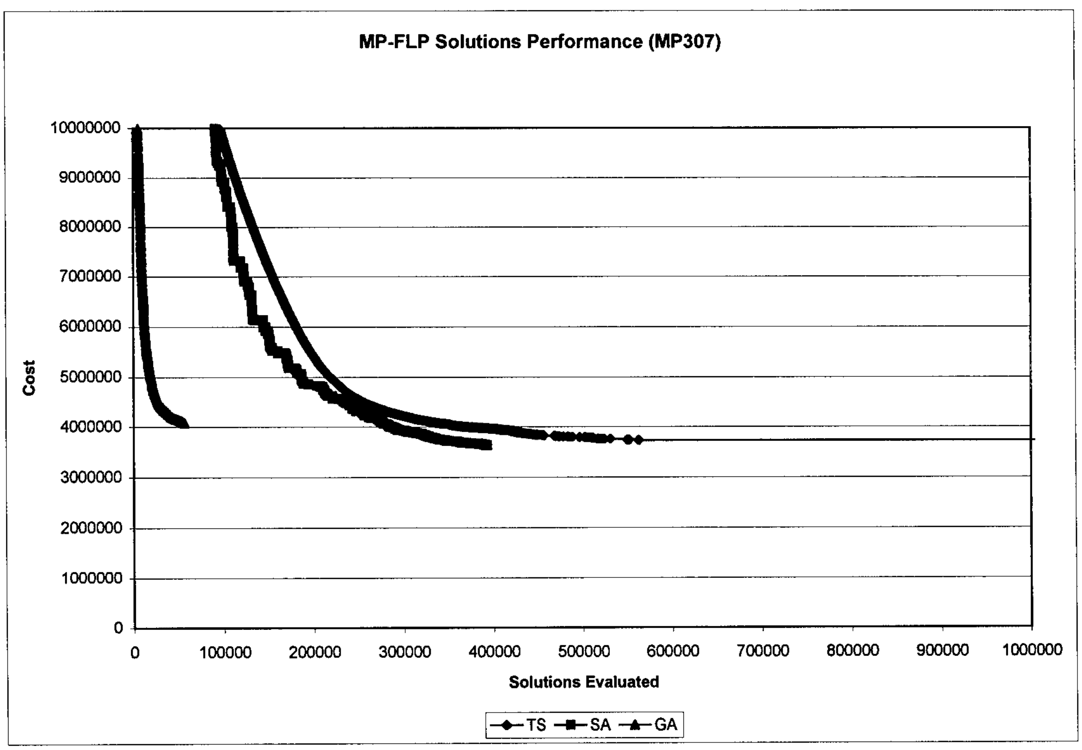

Figure 253. MP-FLP Solutions Performance Close-up (MP307)

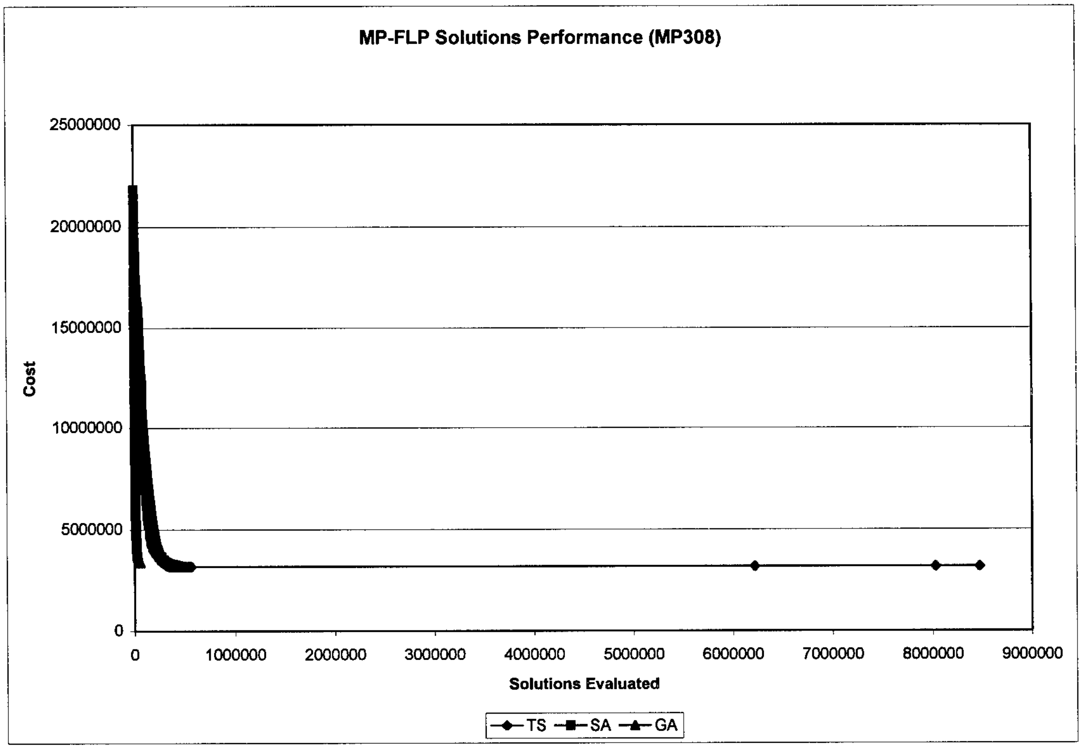

Figure 254. MP-FLP Solutions Performance (MP308)

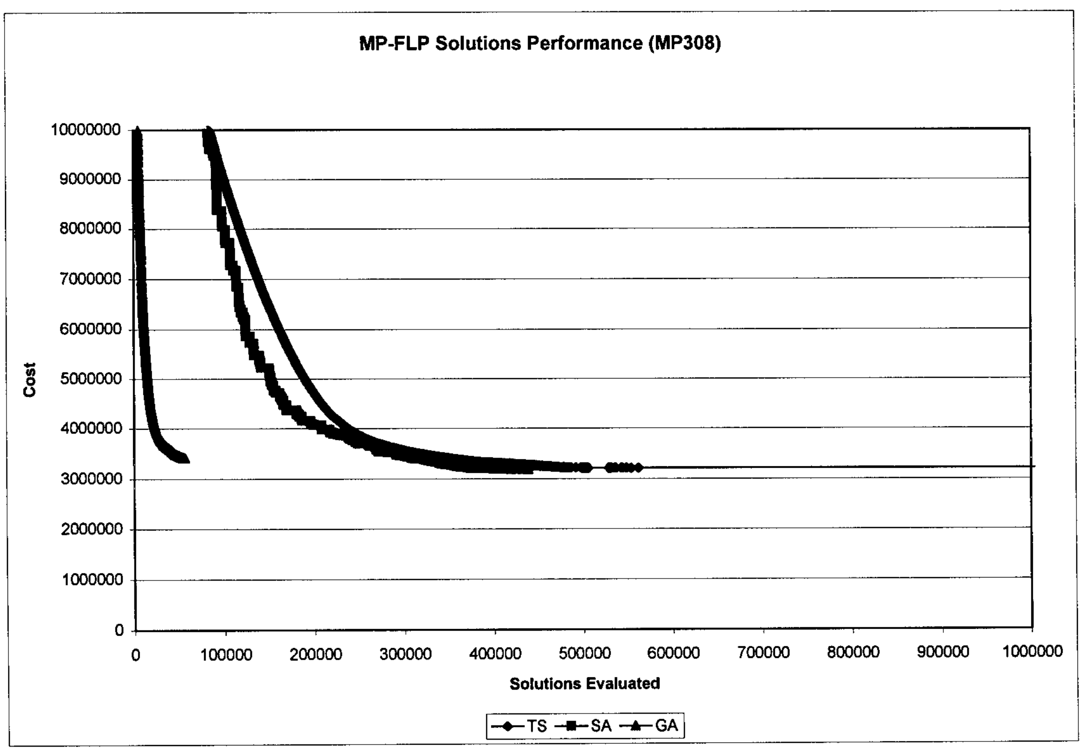

Figure 255. MP-FLP Solutions Performance Close-up (MP308)

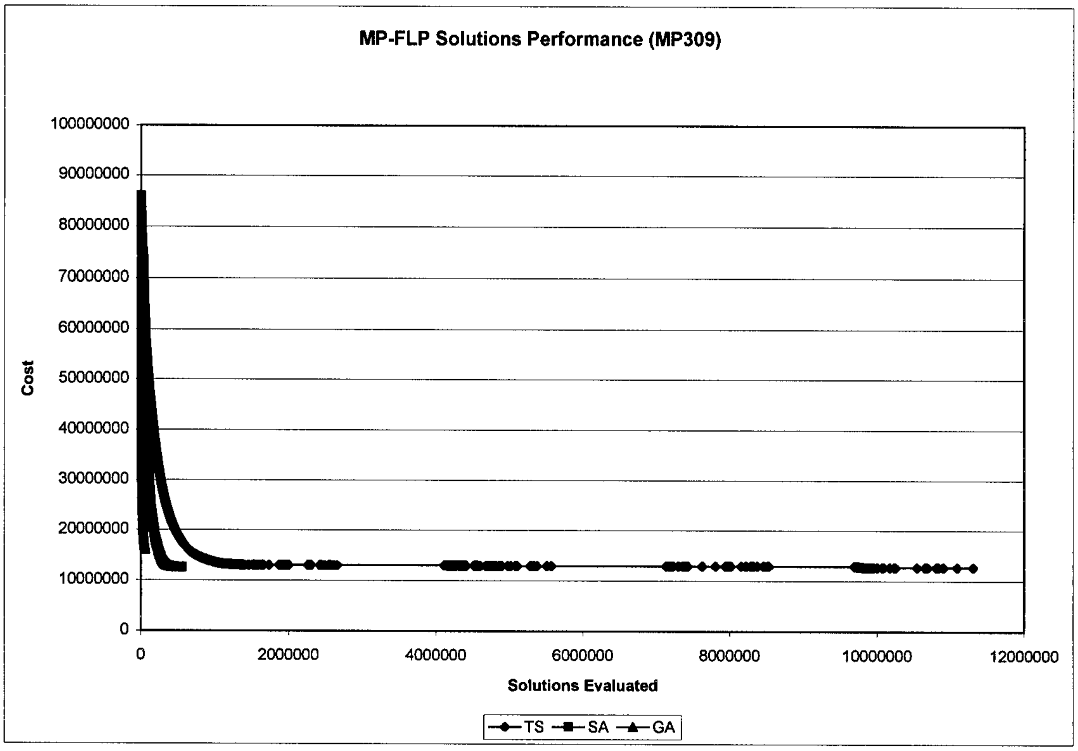

Figure 256. MP-FLP Solutions Performance (MP309)

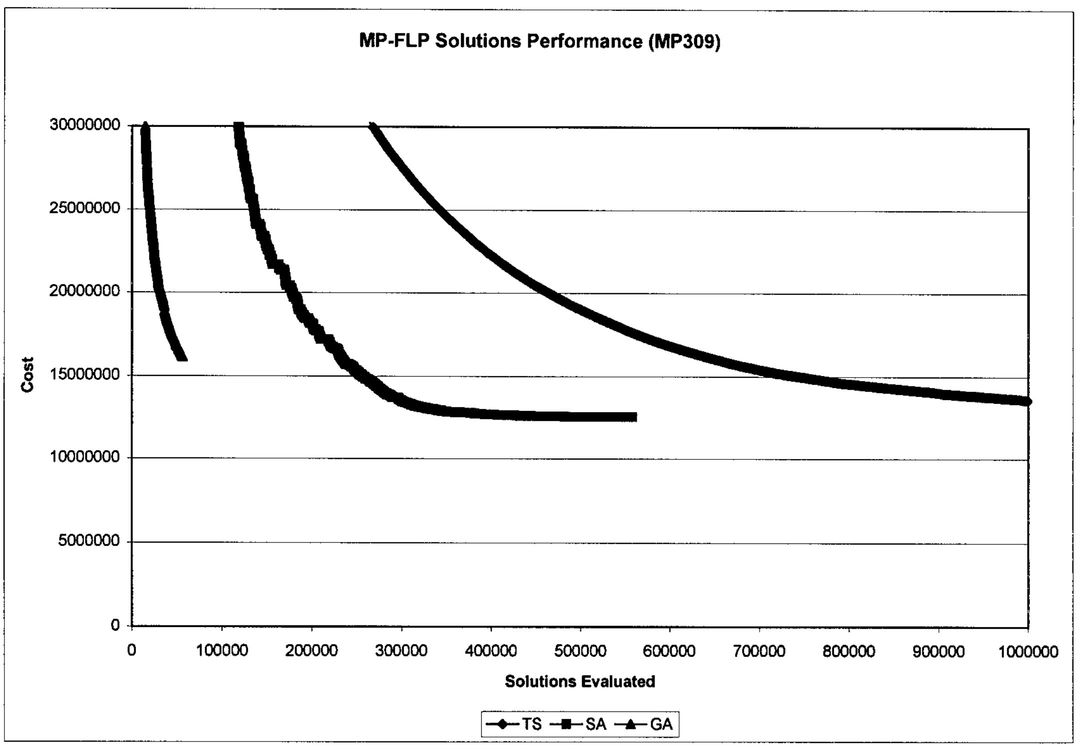

Figure 257. MP-FLP Solutions Performance Close-up (MP309)

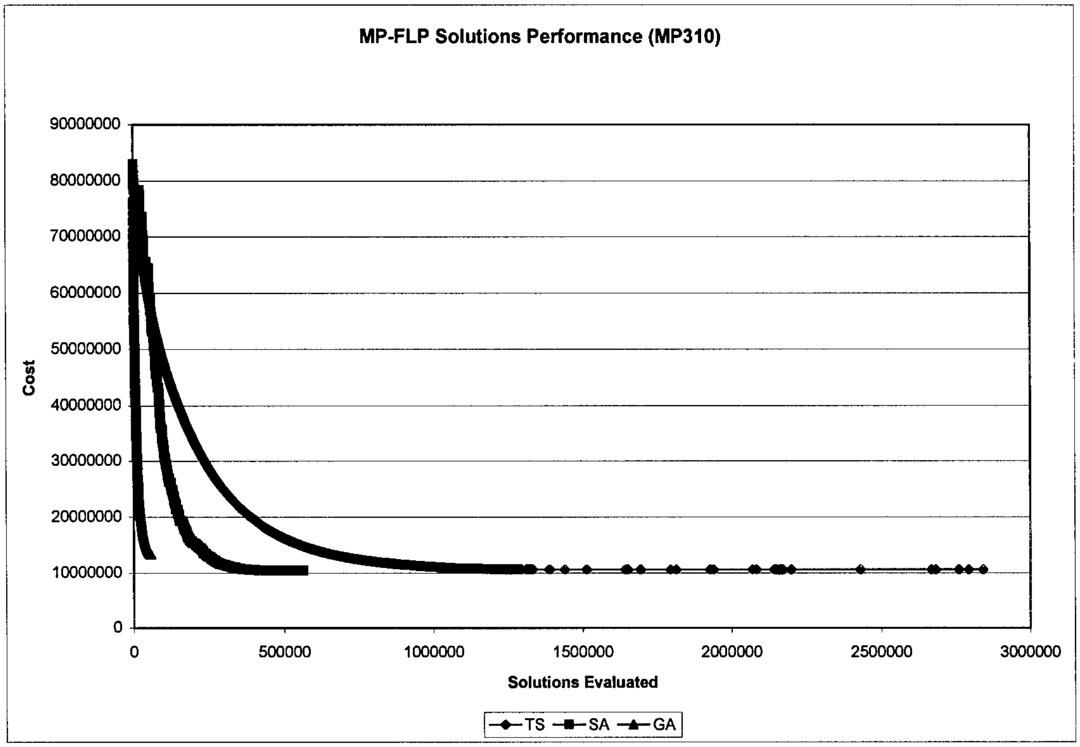

Figure 258. MP-FLP Solutions Performance (MP310)

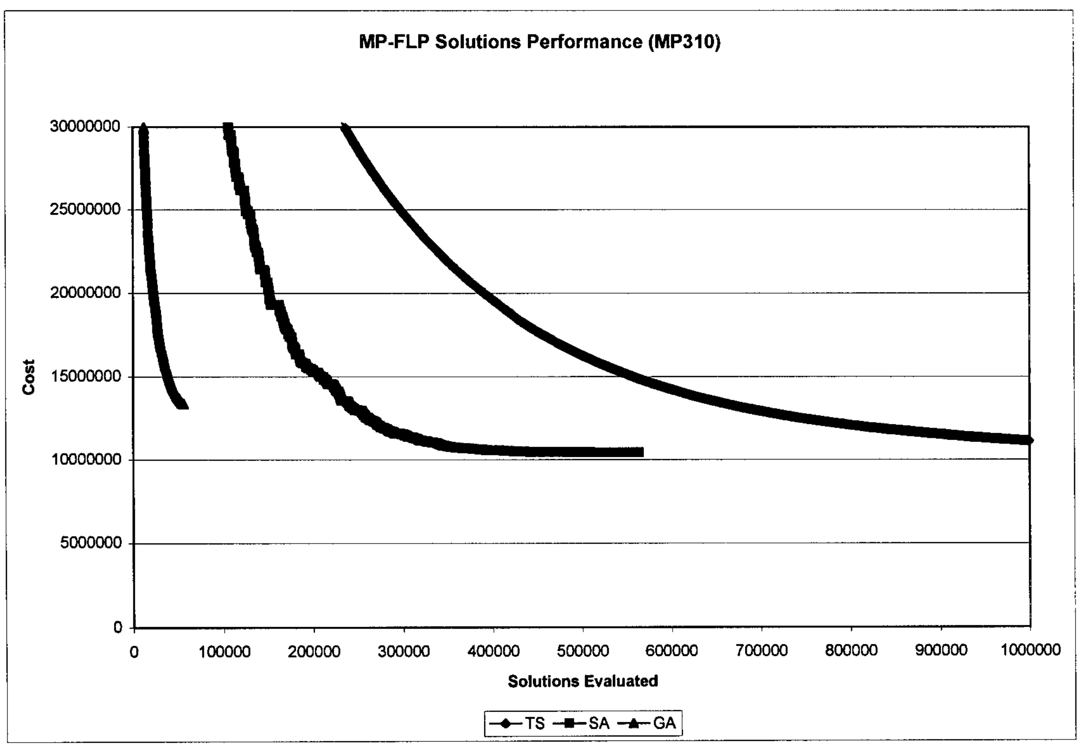

Figure 259. MP-FLP Solutions Performance Close-up (MP310)

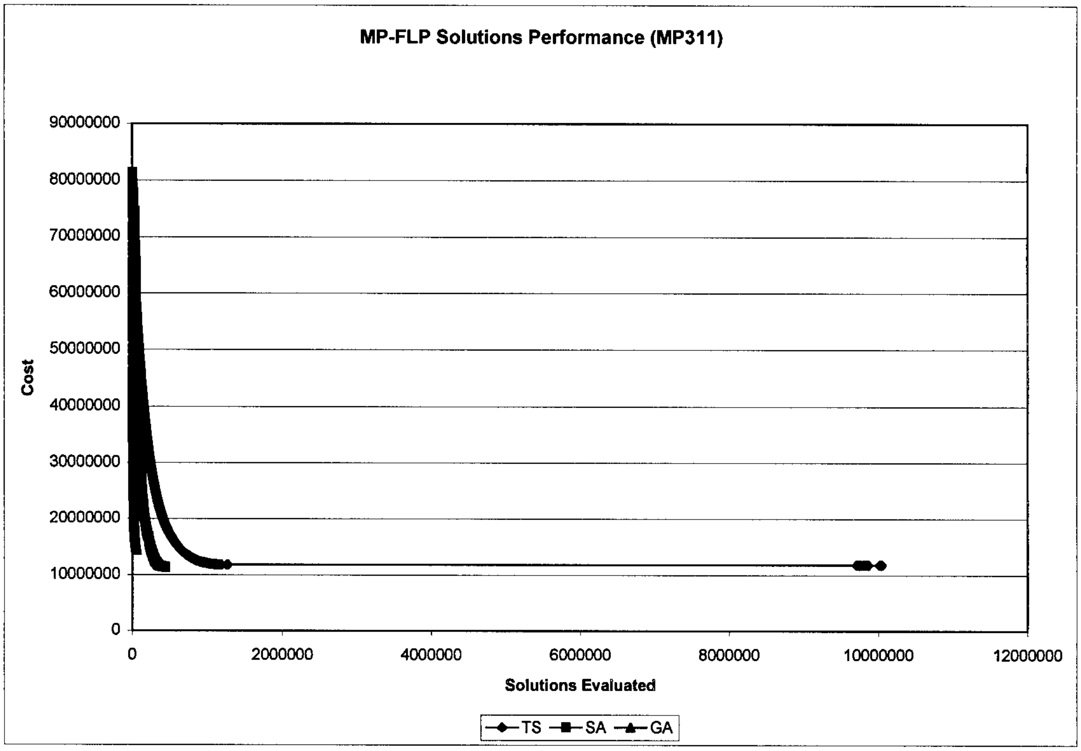

Figure 260. MP-FLP Solutions Performance (MP311)

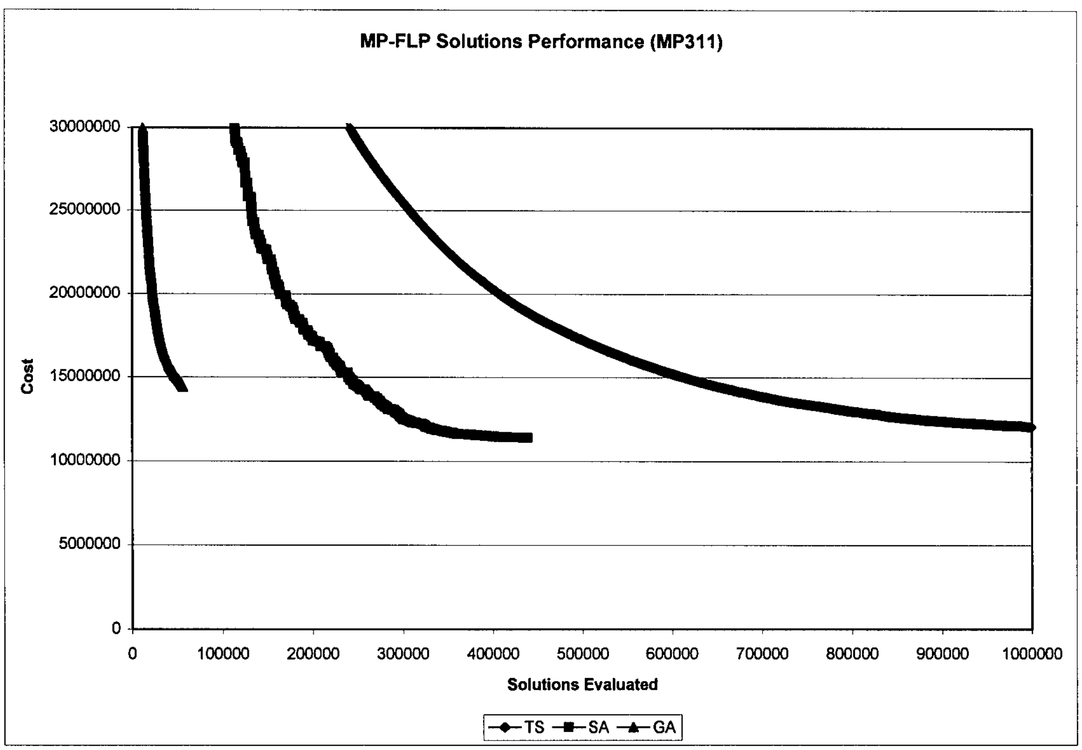

Figure 261. MP-FLP Solutions Performance Close-up (MP311)

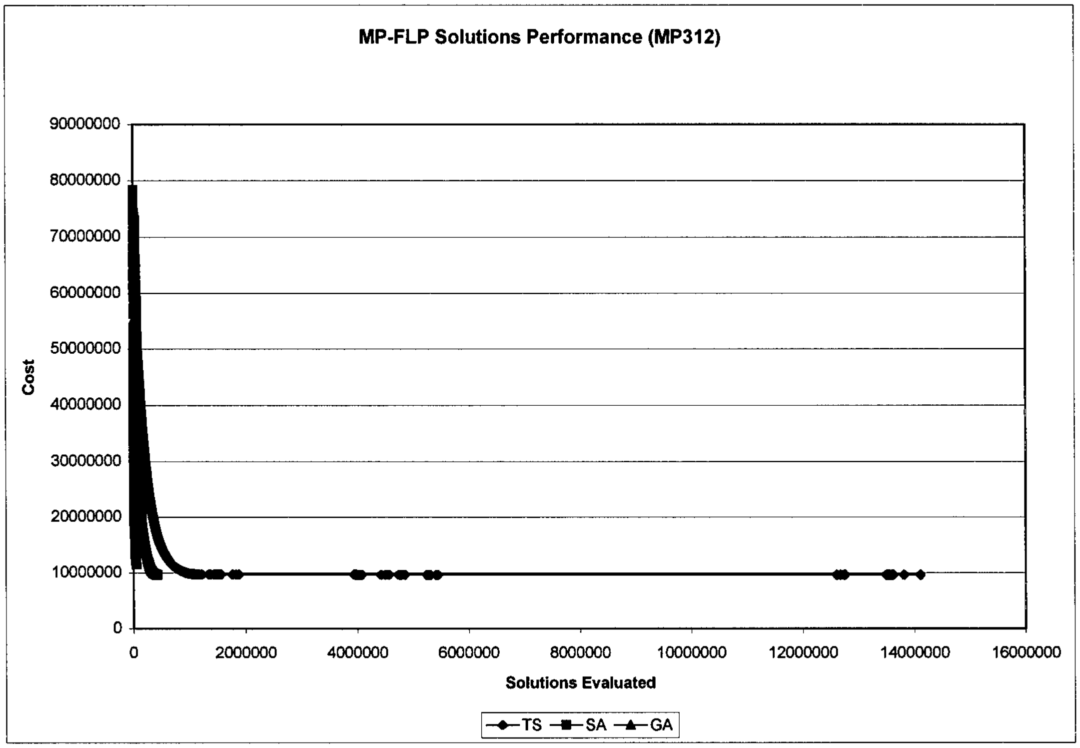

Figure 262. MP-FLP Solutions Performance (MP312)

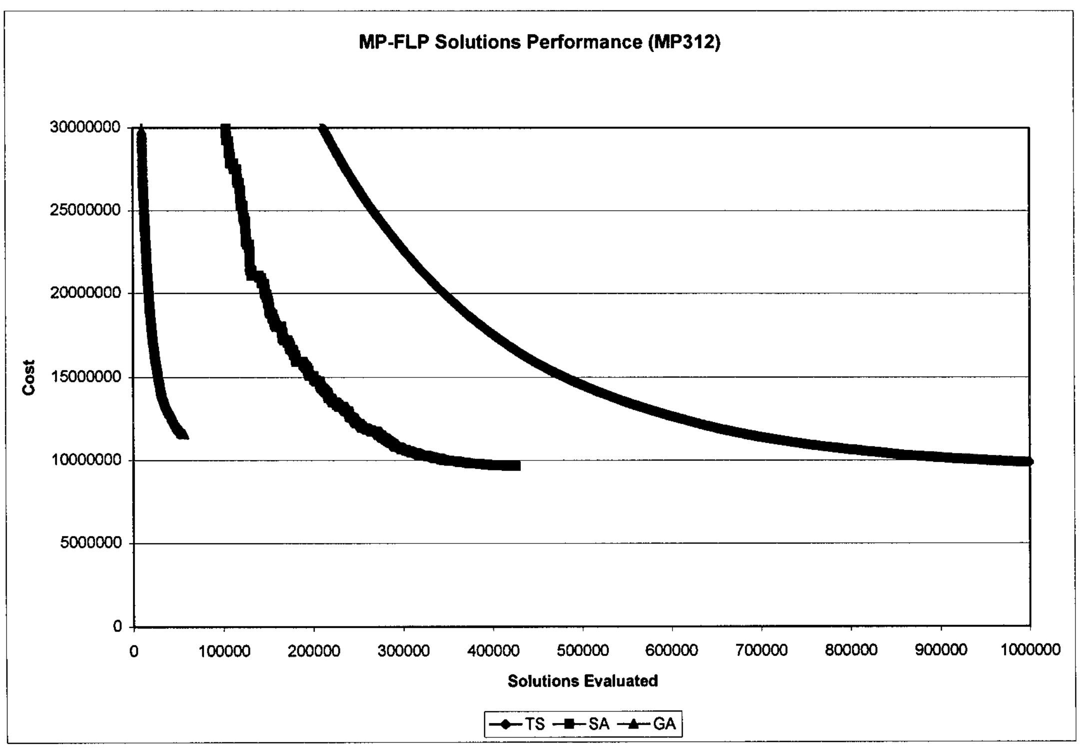

Figure 263. MP-FLP Solutions Performance Close-up (MP312)

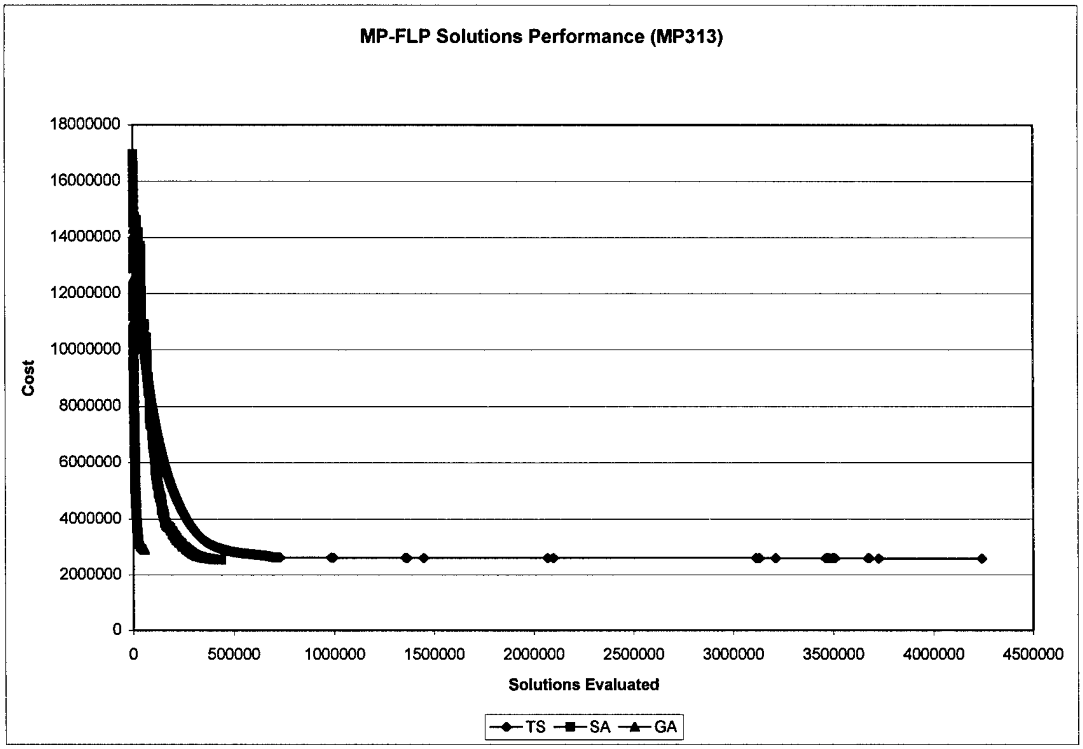

Figure 264. MP-FLP Solutions Performance (MP313)

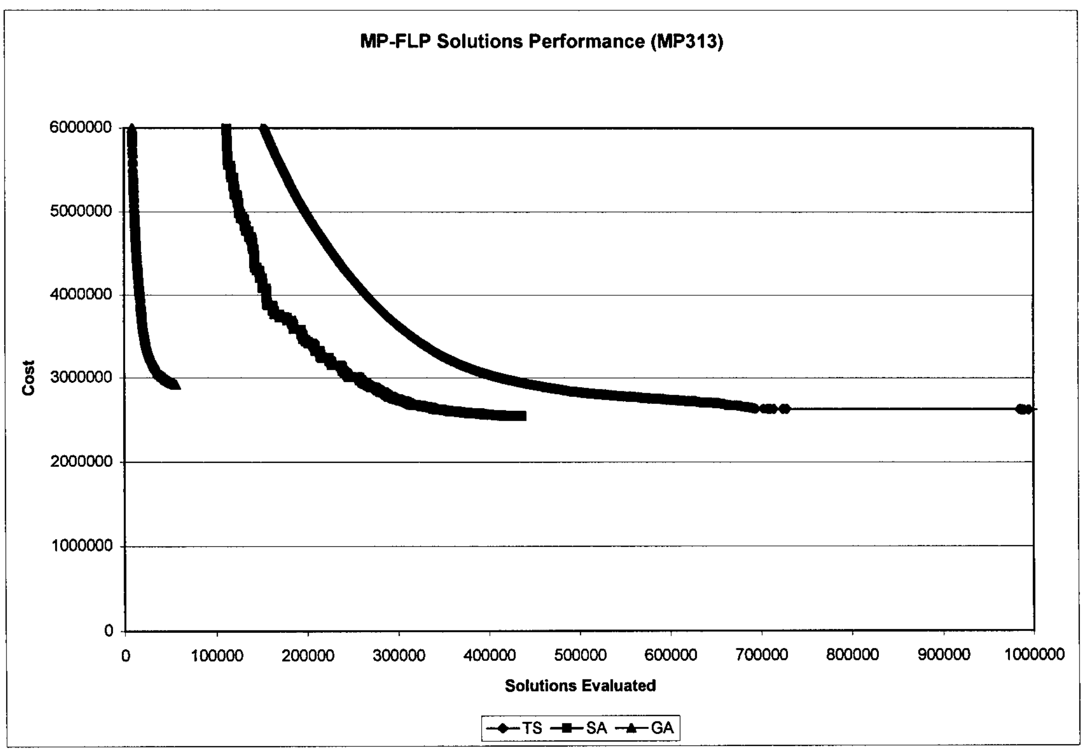

Figure 265. MP-FLP Solutions Performance Close-up (MP313)

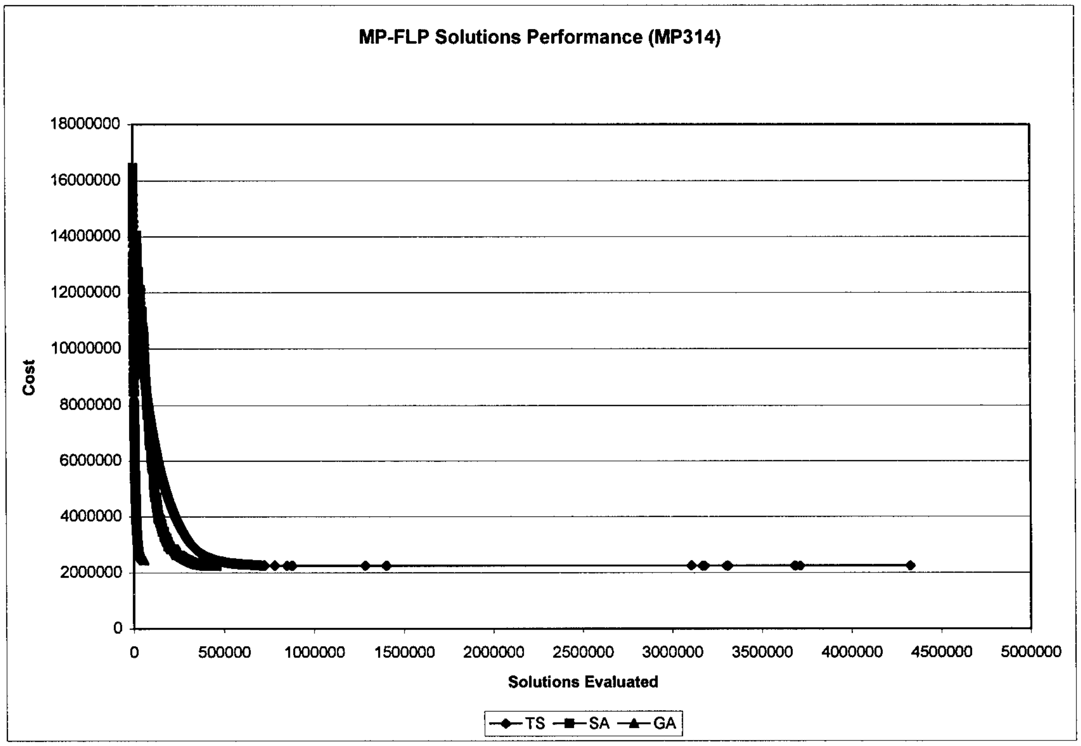

Figure 266. MP-FLP Solutions Performance (MP314)

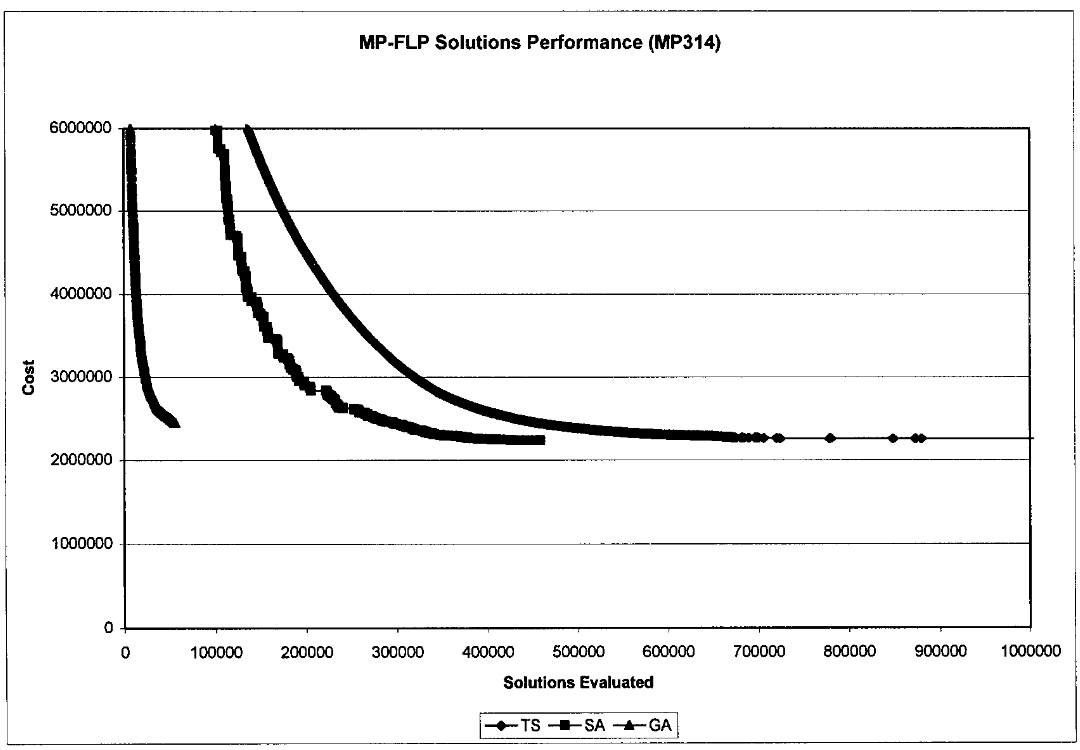

Figure 267. MP-FLP Solutions Performance Close-up (MP314)

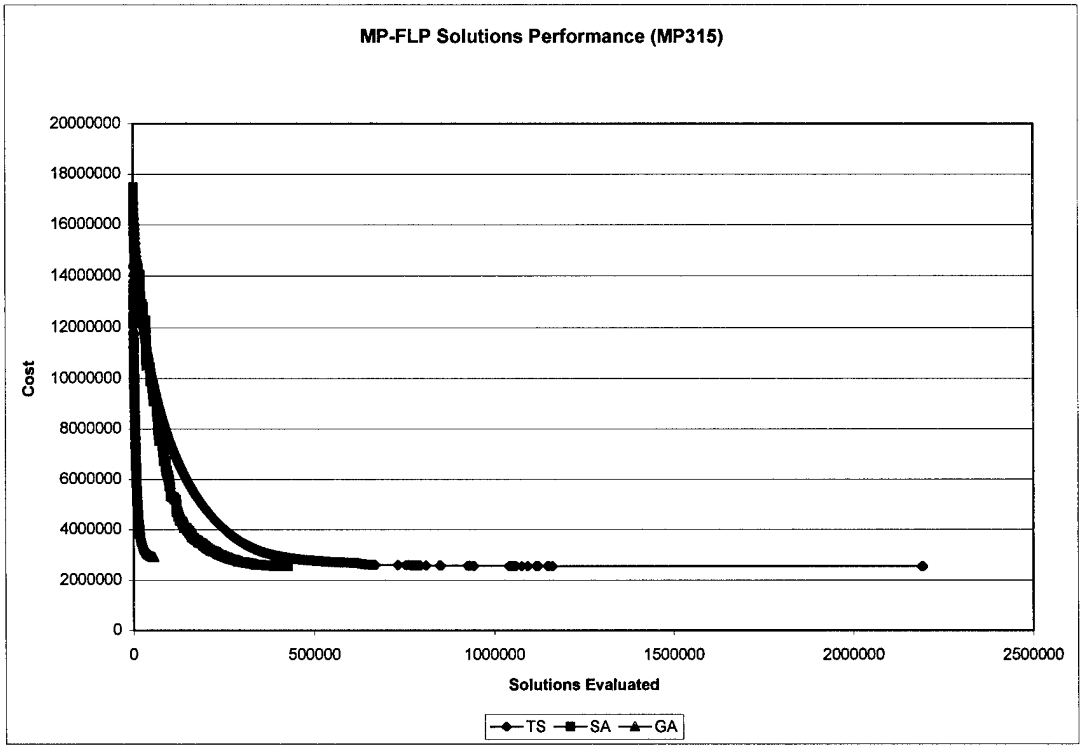

Figure 268. MP-FLP Solutions Performance (MP315)

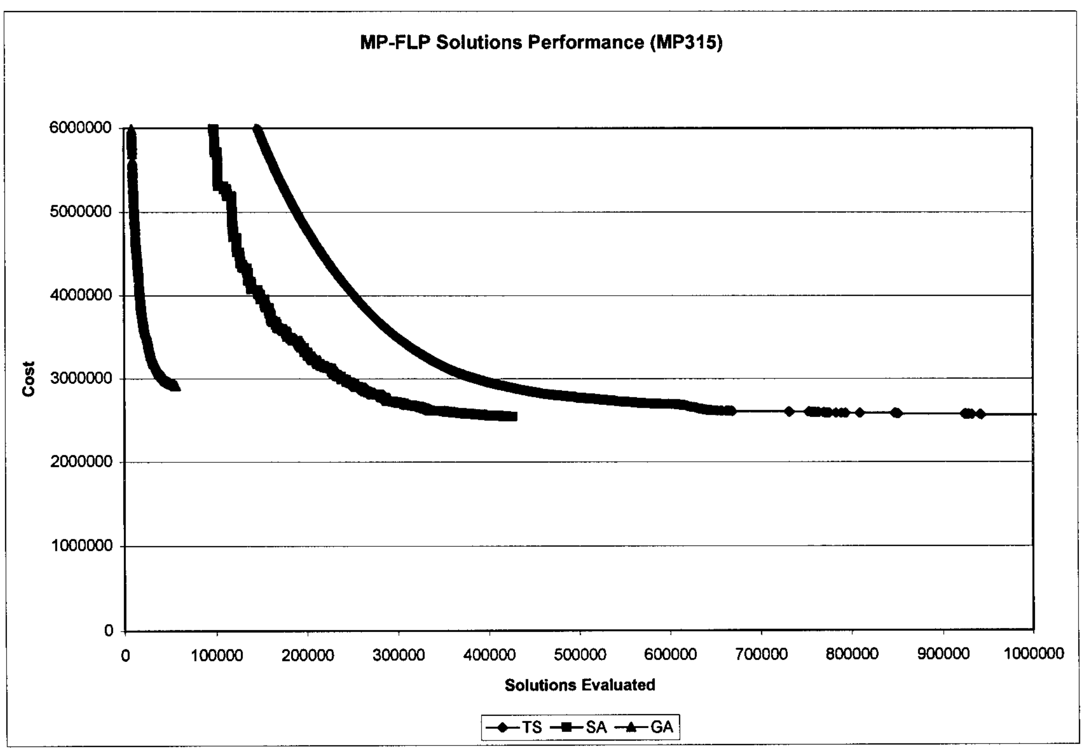

Figure 269. MP-FLP Solutions Performance Close-up (MP315)

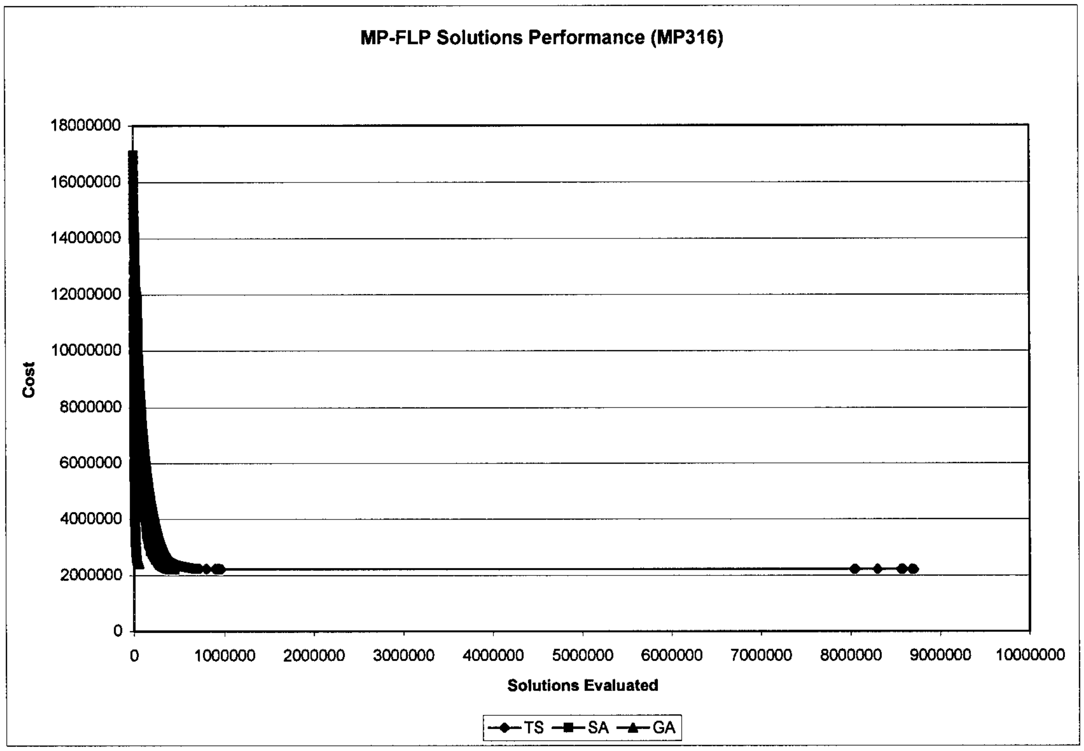

Figure 270. MP-FLP Solutions Performance (MP316)

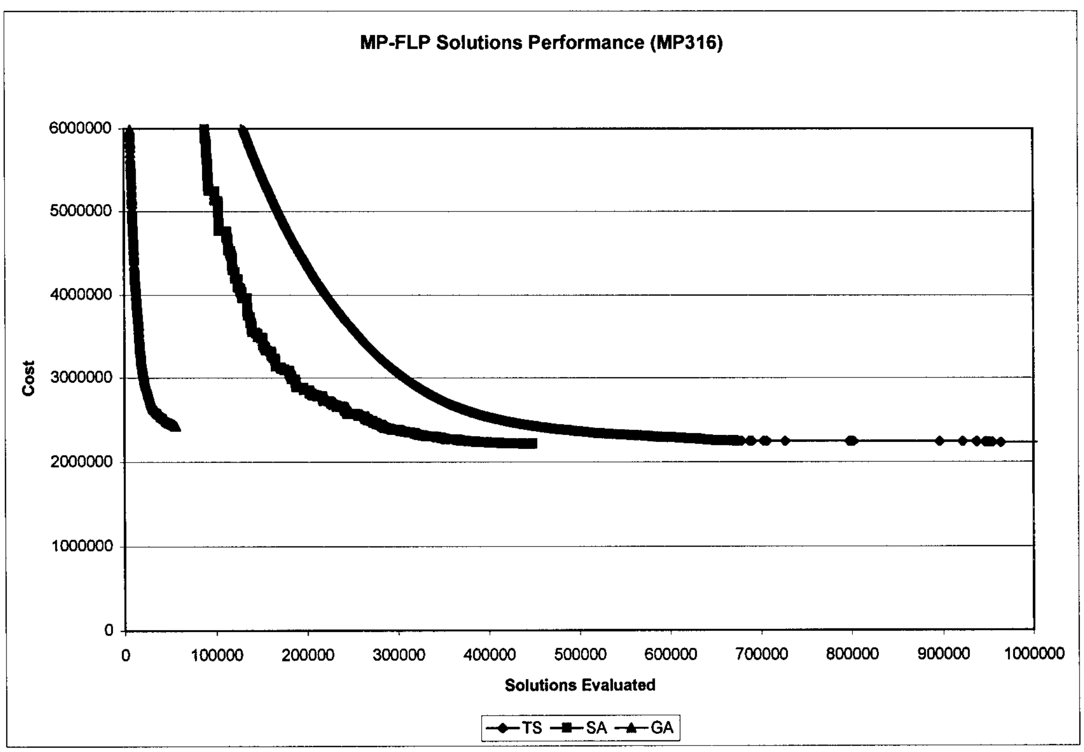

Figure 271. MP-FLP Solutions Performance Close-up (MP316)

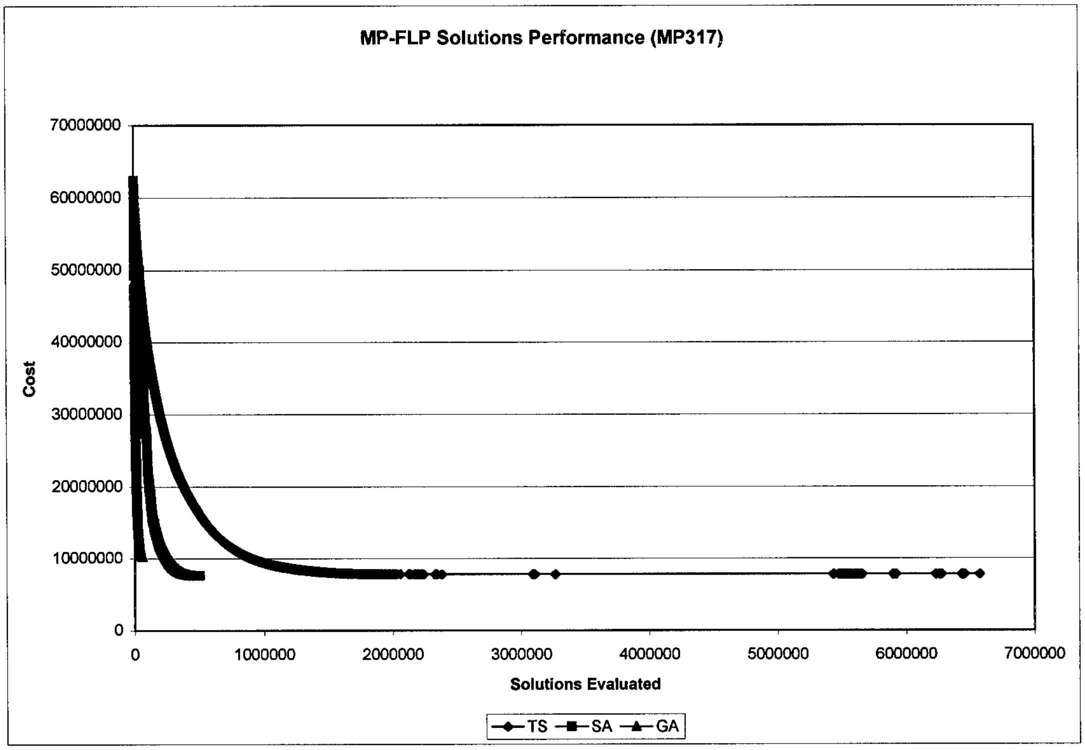

Figure 272. MP-FLP Solutions Performance (MP317)

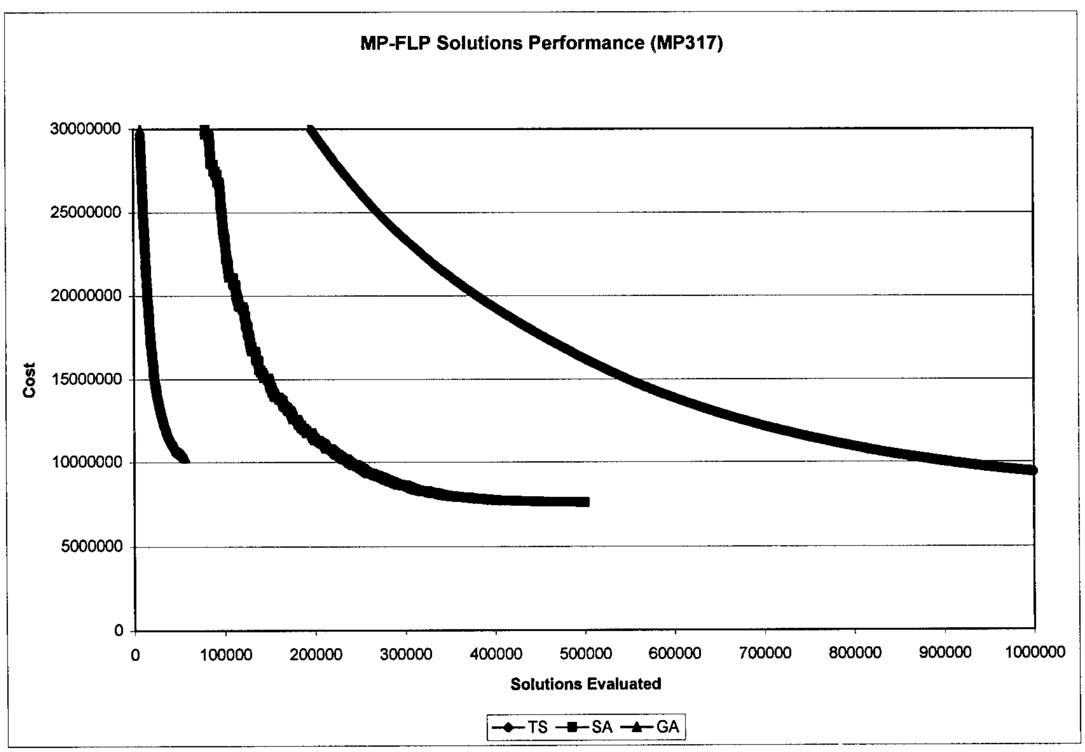

Figure 273. MP-FLP Solutions Performance Close-up (MP317)

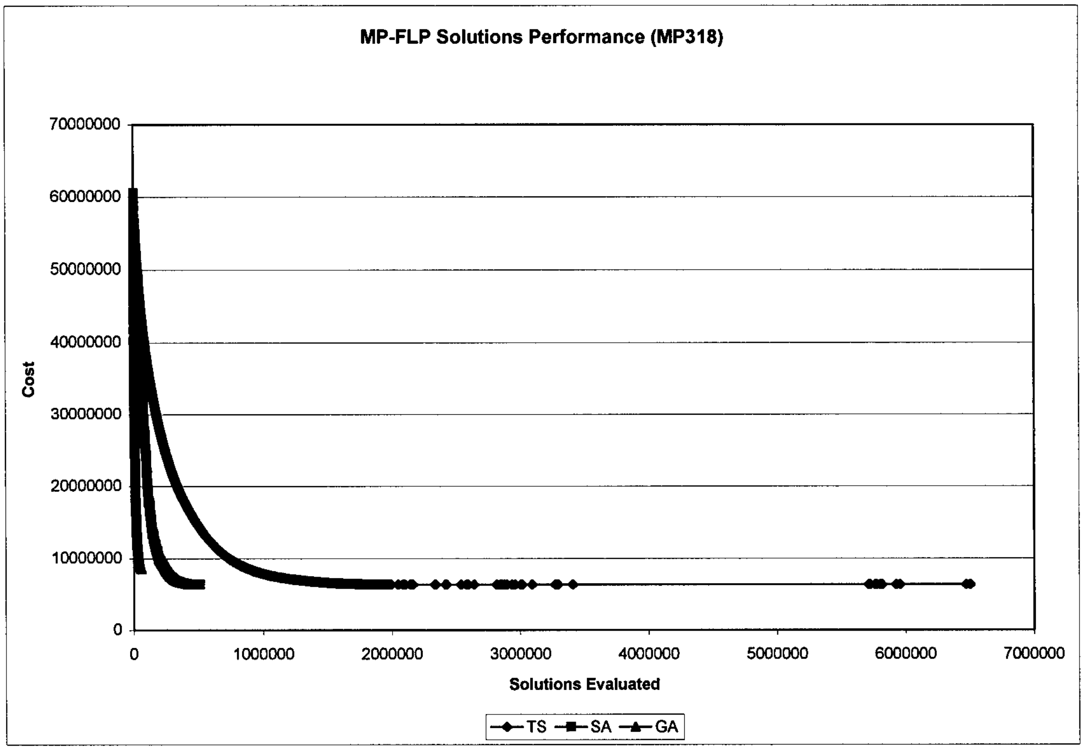

Figure 274. MP-FLP Solutions Performance (MP318)

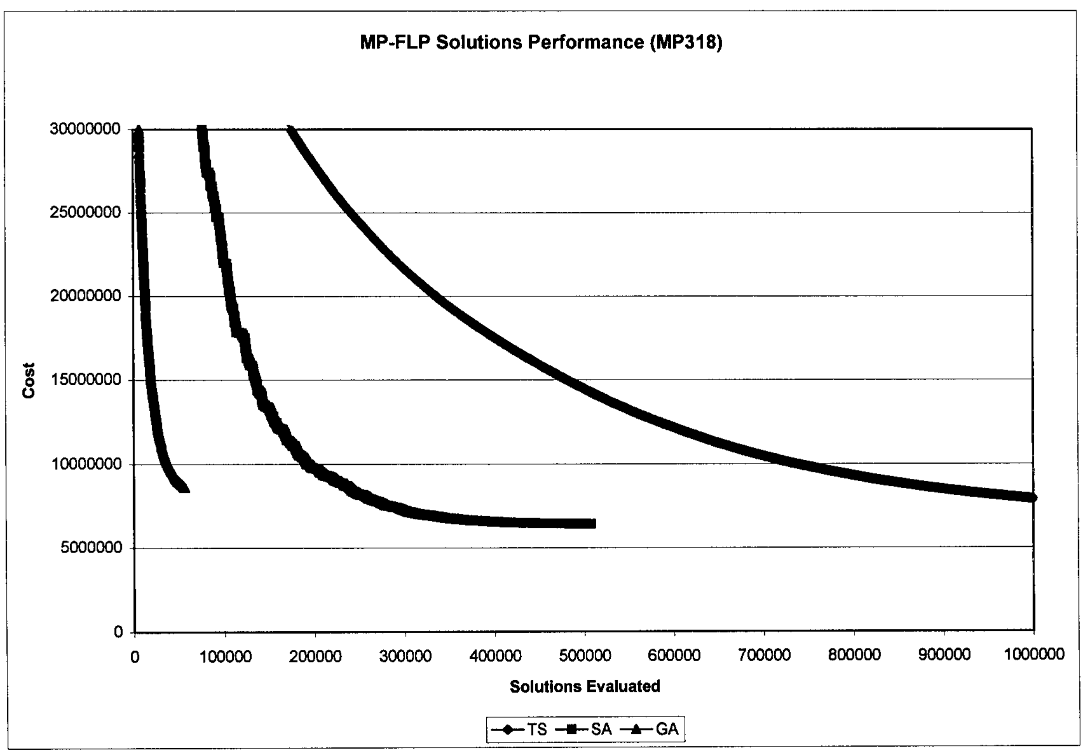

Figure 275. MP-FLP Solutions Performance Close-up (MP318)

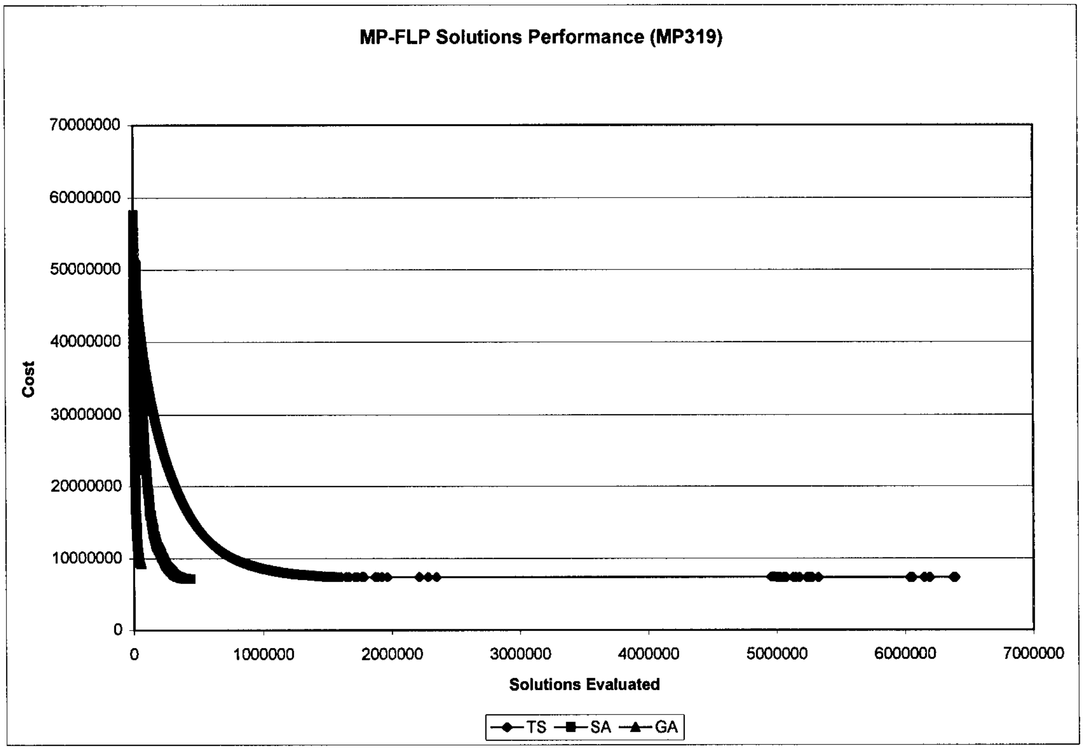

Figure 276. MP-FLP Solutions Performance (MP319)

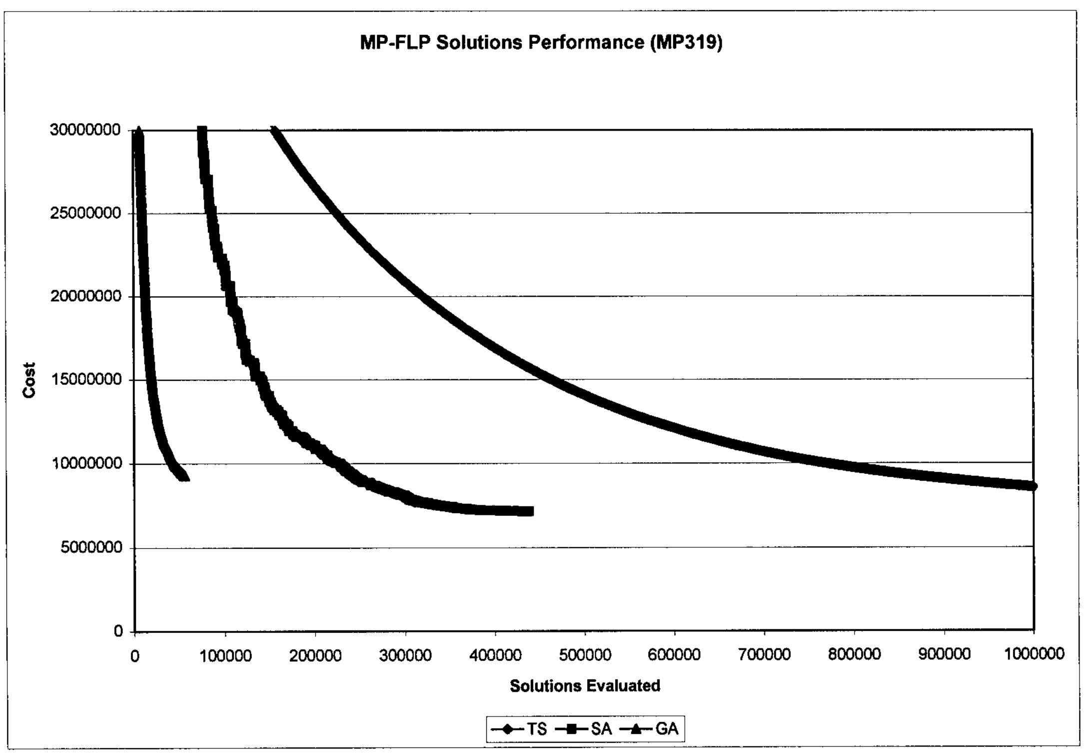

Figure 277. MP-FLP Solutions Performance Close-up (MP319)

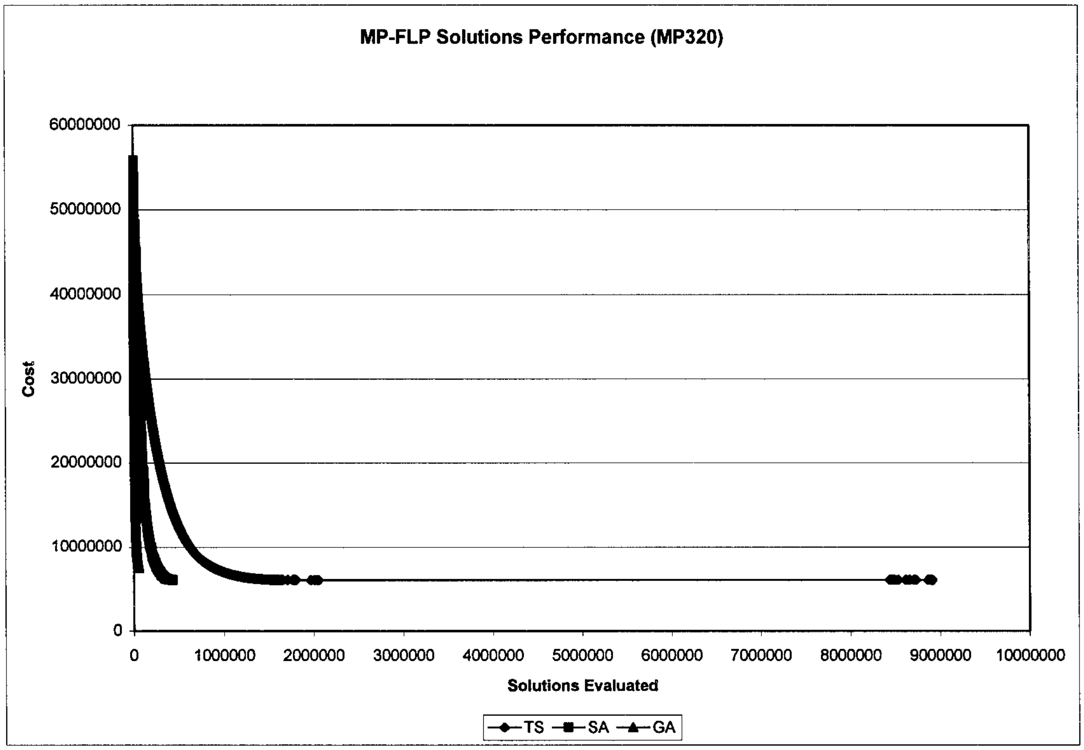

Figure 278. MP-FLP Solutions Performance (MP320)

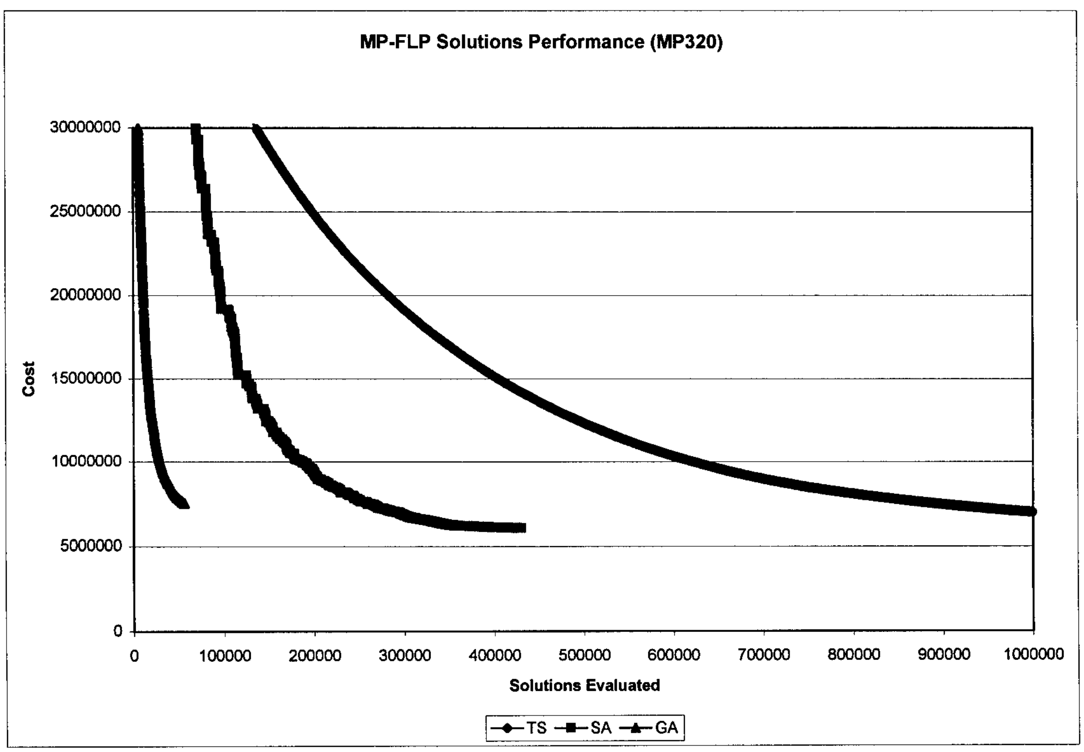

Figure 279. MP-FLP Solutions Performance Close-up (MP320)

APPENDIX I

MC-FLP TIME PERFORMANCE CHARTS

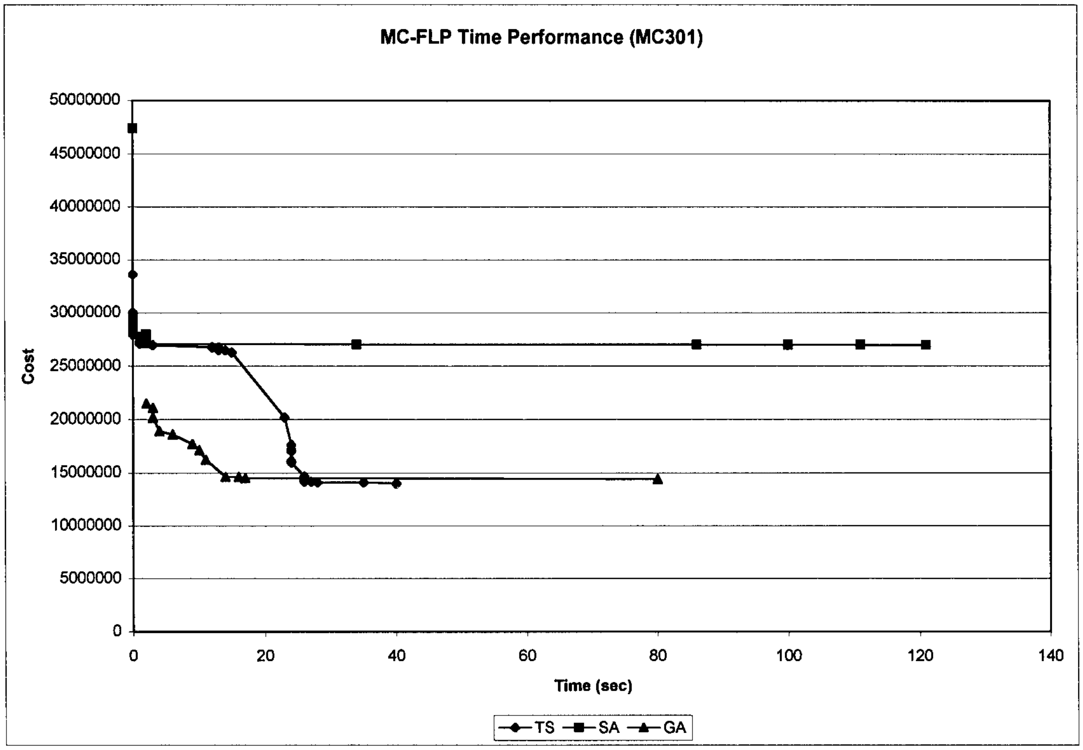

Figure 280. MC-FLP Time Performance (MC301)

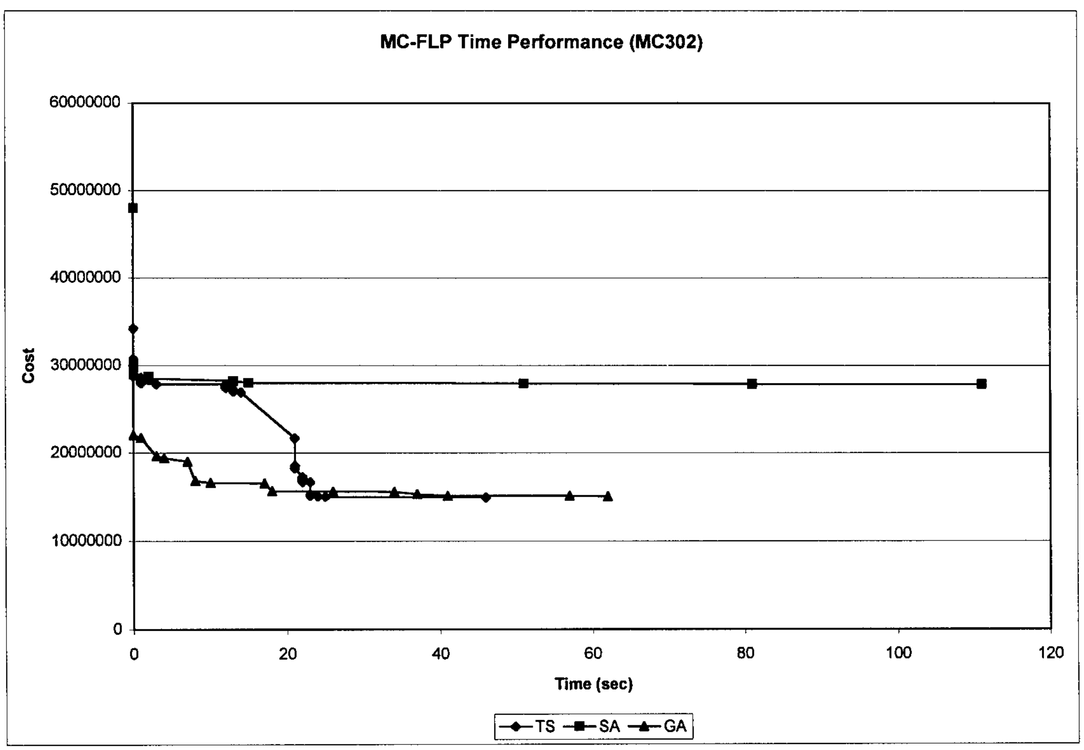

Figure 281. MC-FLP Time Performance (MC302)

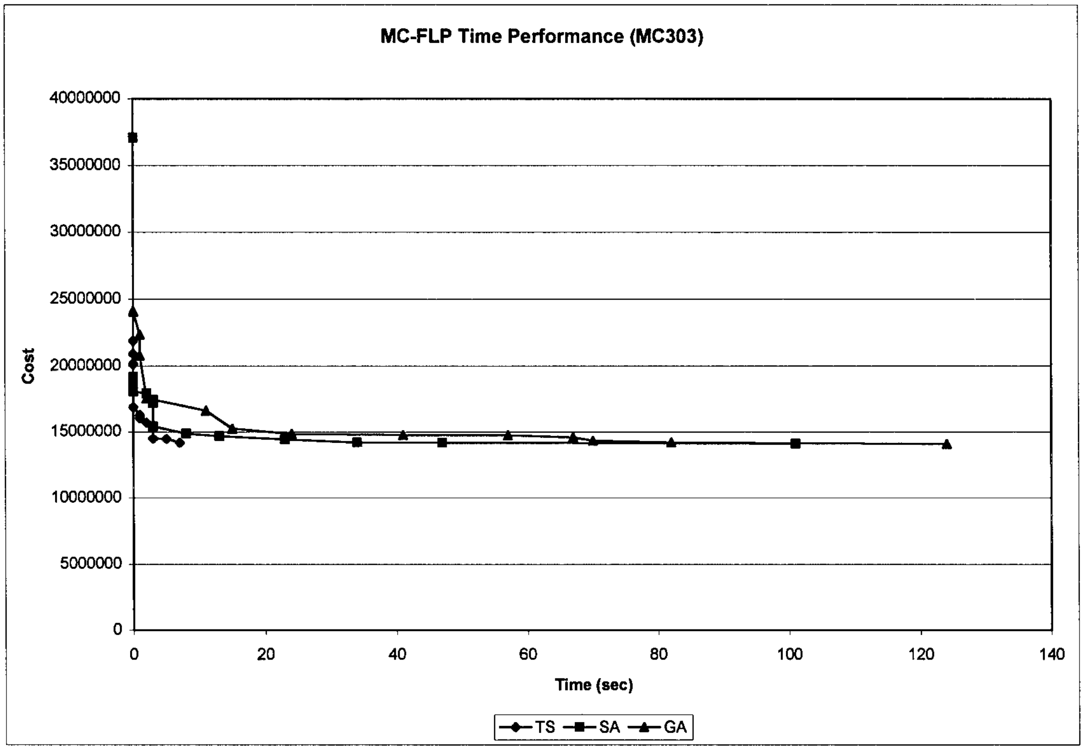

Figure 282. MC-FLP Time Performance (MC303)

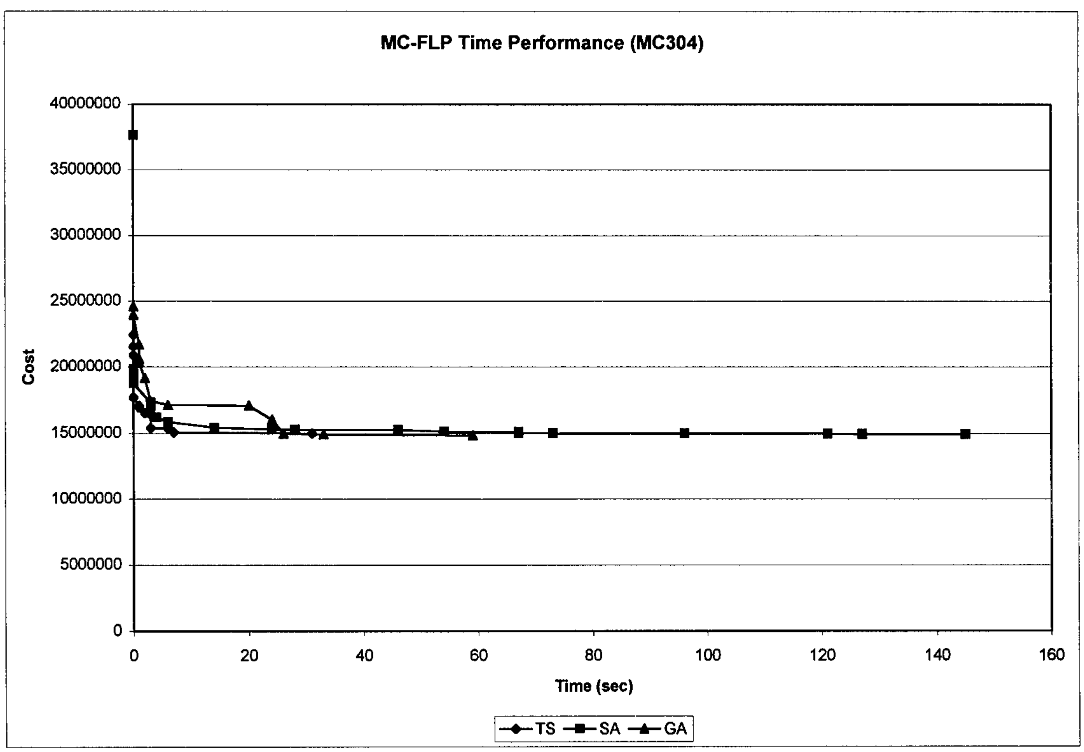

Figure 283. MC-FLP Time Performance (MC304)

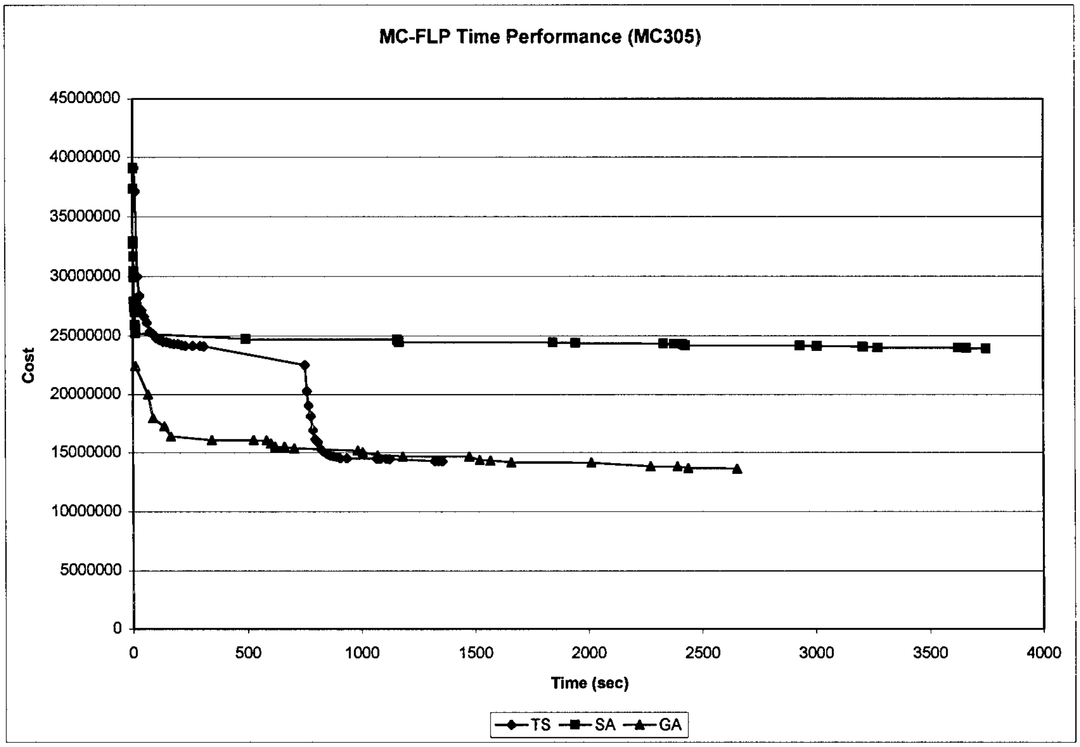

Figure 284. MC-FLP Time Performance (MC305)

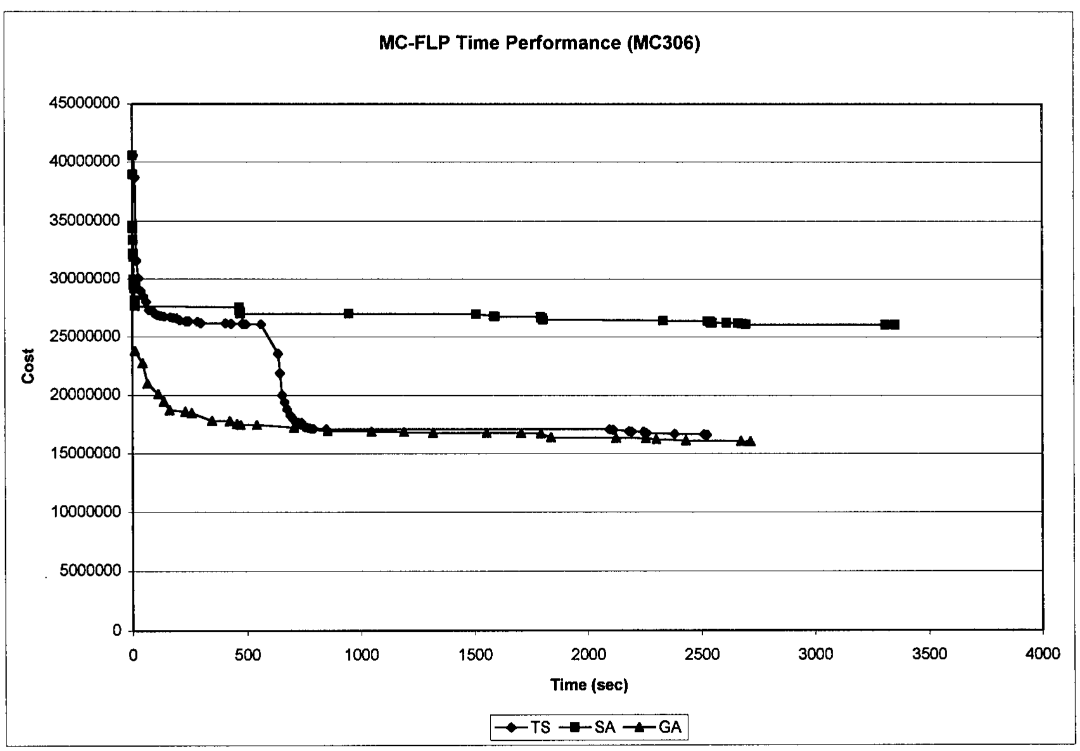

Figure 285. MC-FLP Time Performance (MC306)

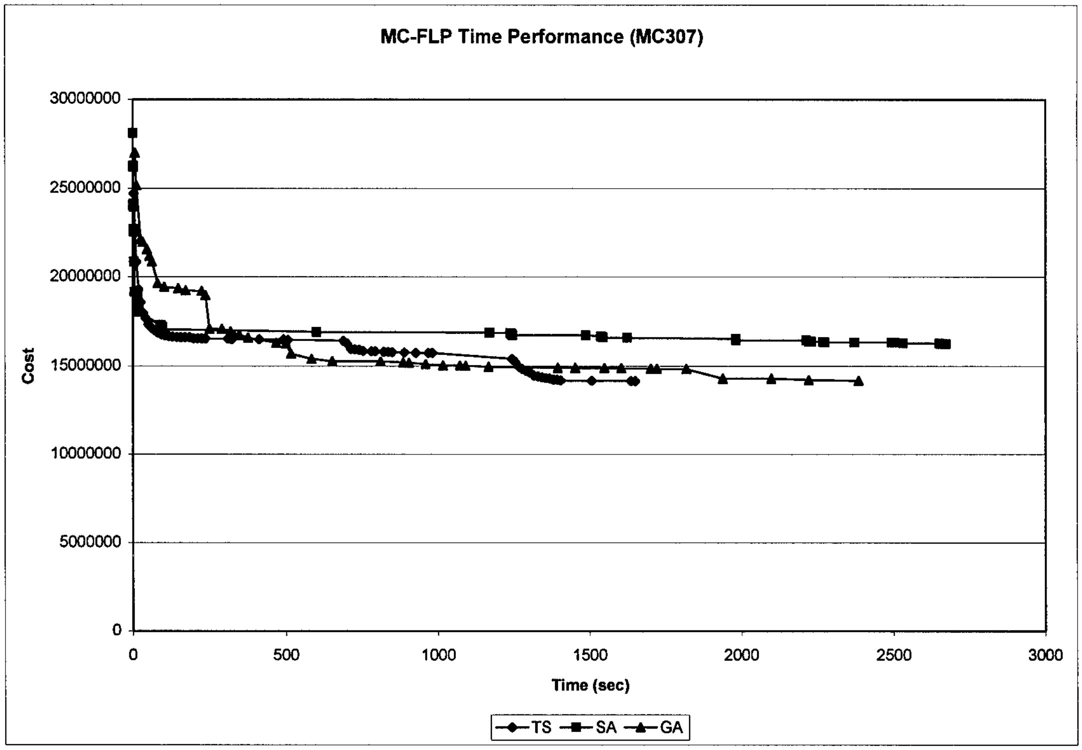

Figure 286. MC-FLP Time Performance (MC307)

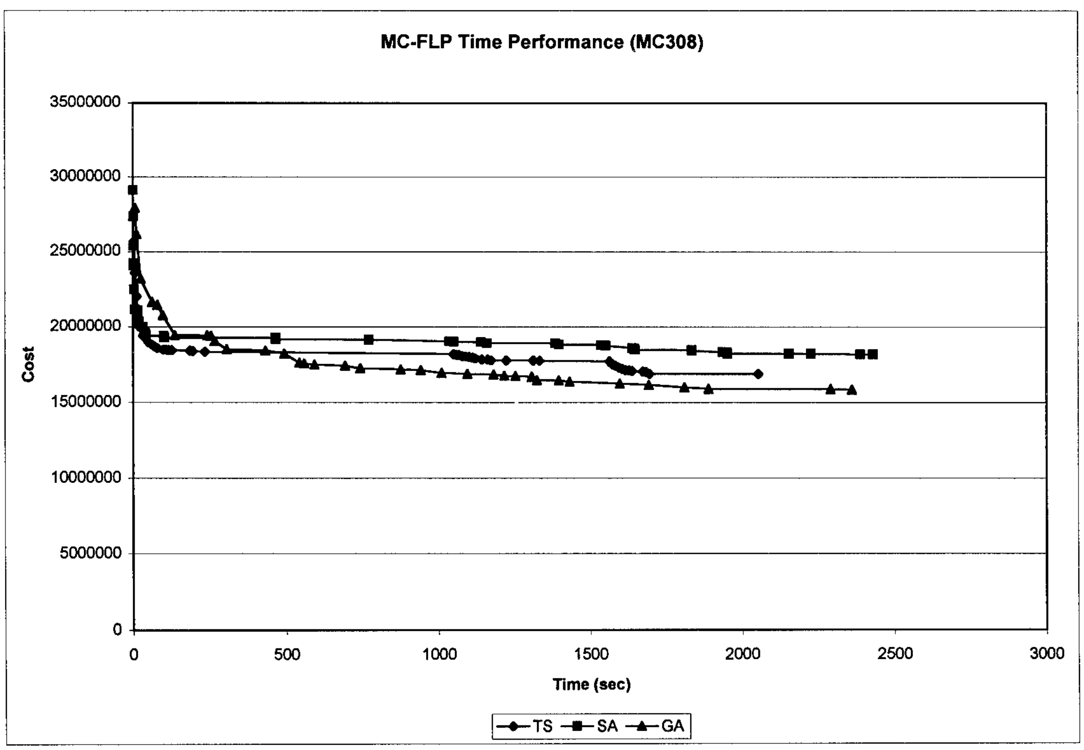

Figure 287. MC-FLP Time Performance (MC308)

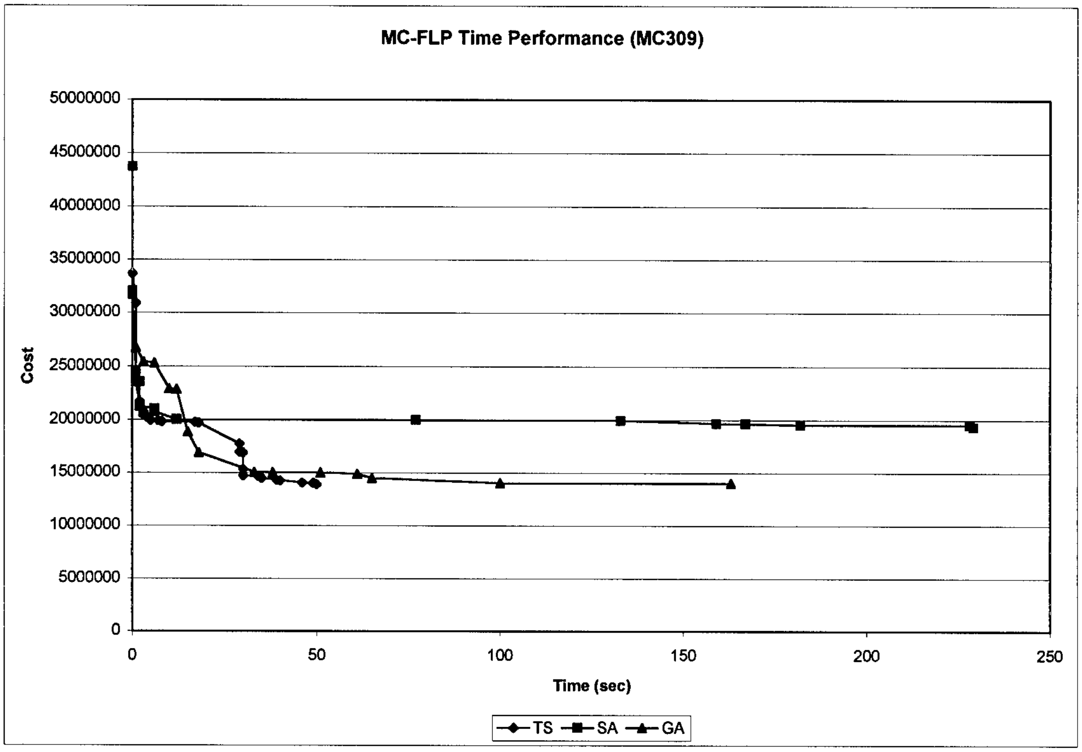

Figure 288. MC-FLP Time Performance (MC309)

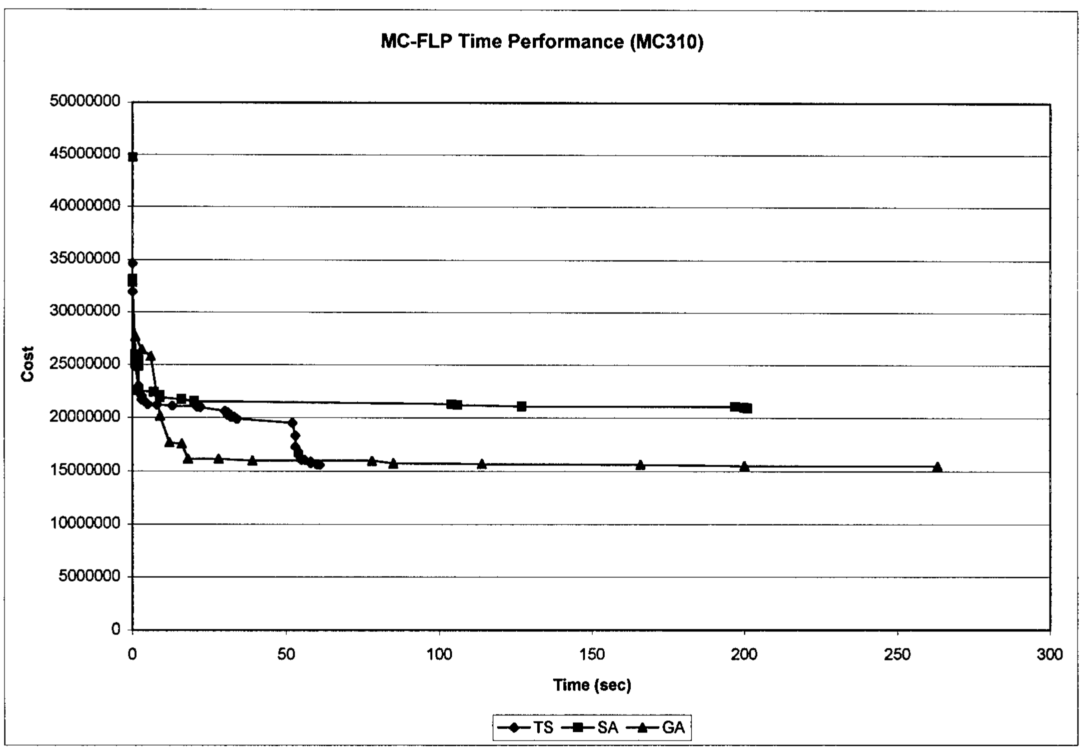

Figure 289. MC-FLP Time Performance (MC310)

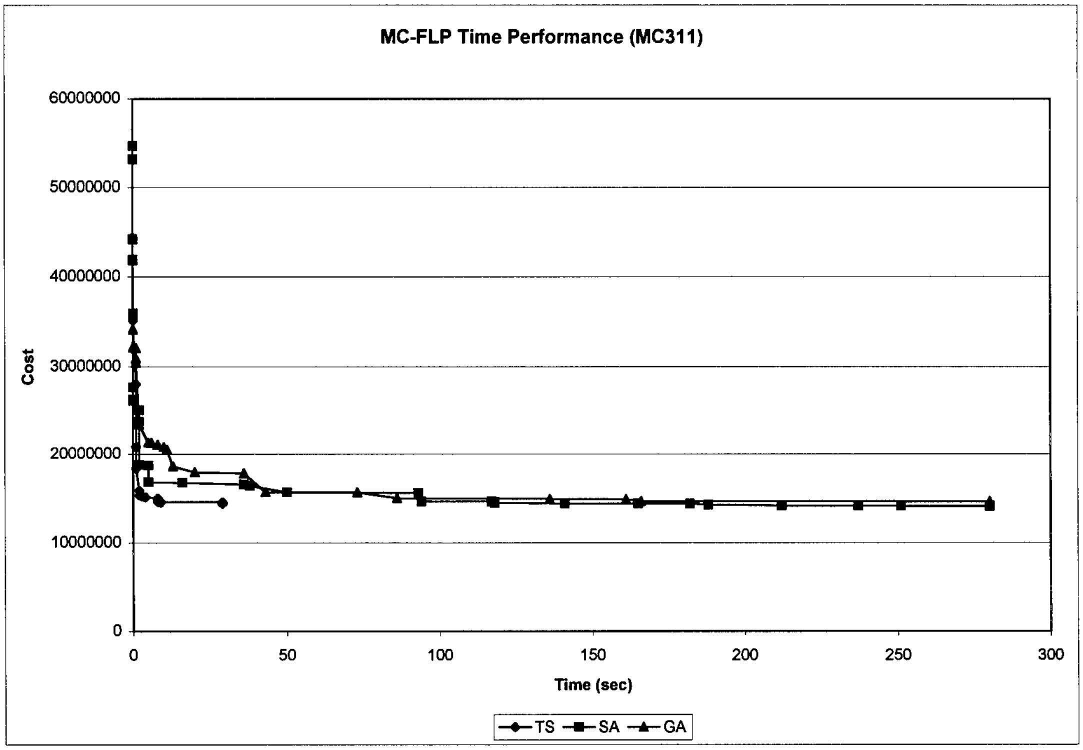

Figure 290. MC-FLP Time Performance (MC311)

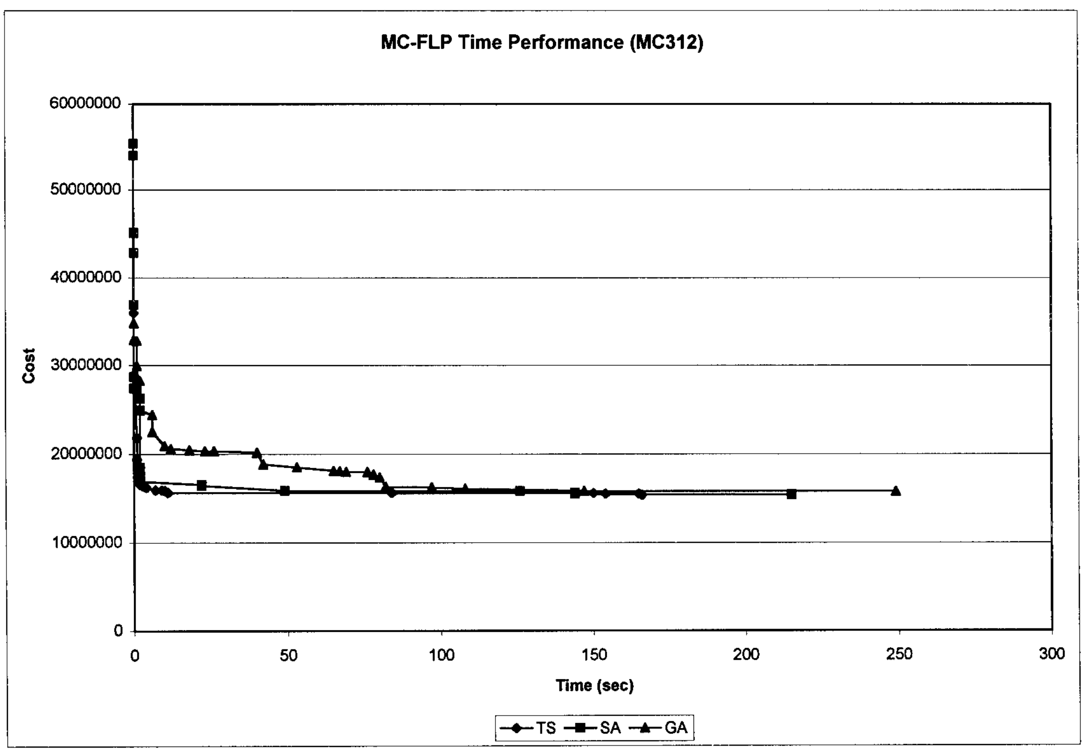

Figure 291. MC-FLP Time Performance (MC312)

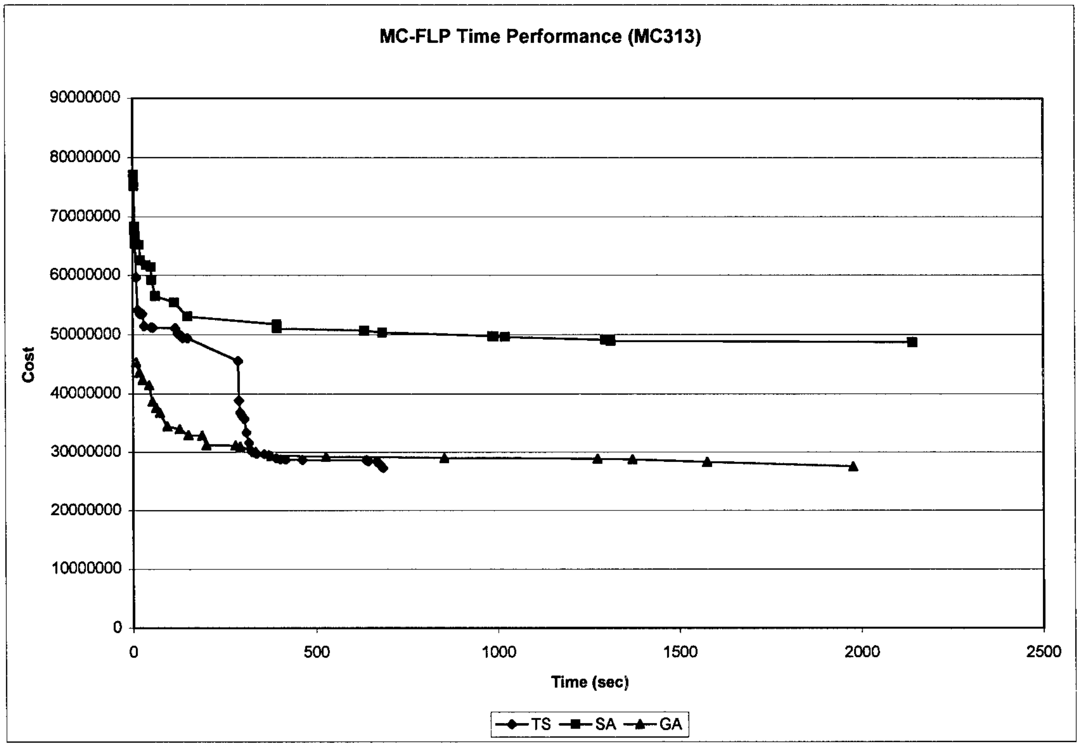

Figure 292. MC-FLP Time Performance (MC313)

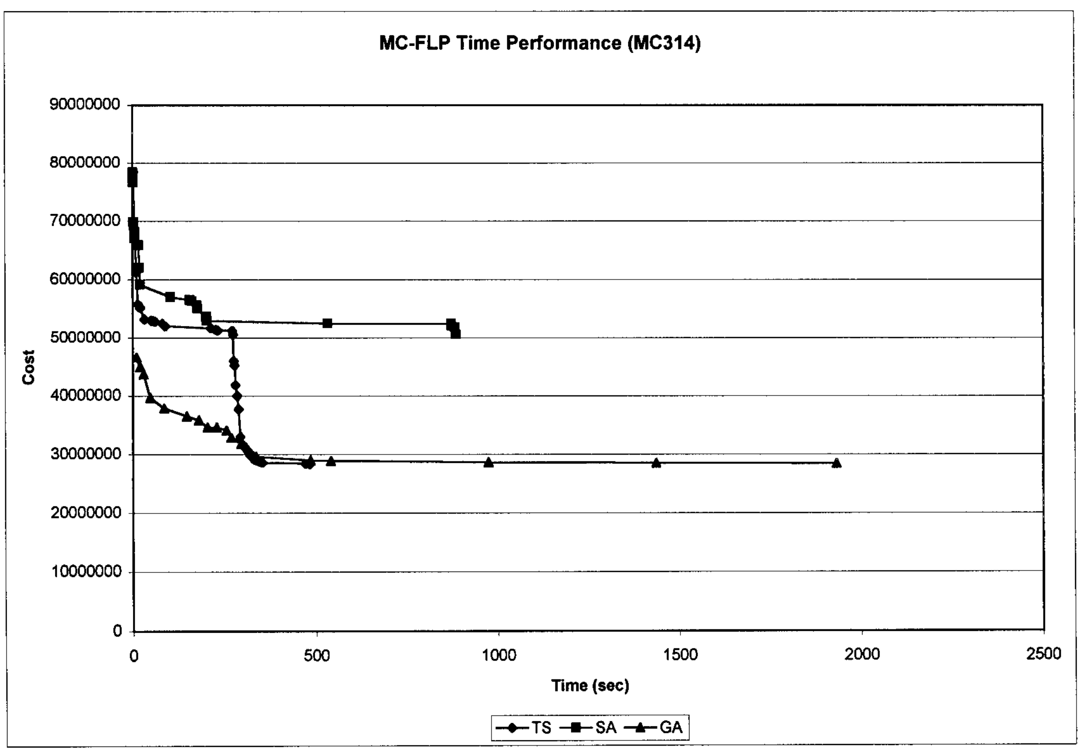

Figure 293. MC-FLP Time Performance (MC314)

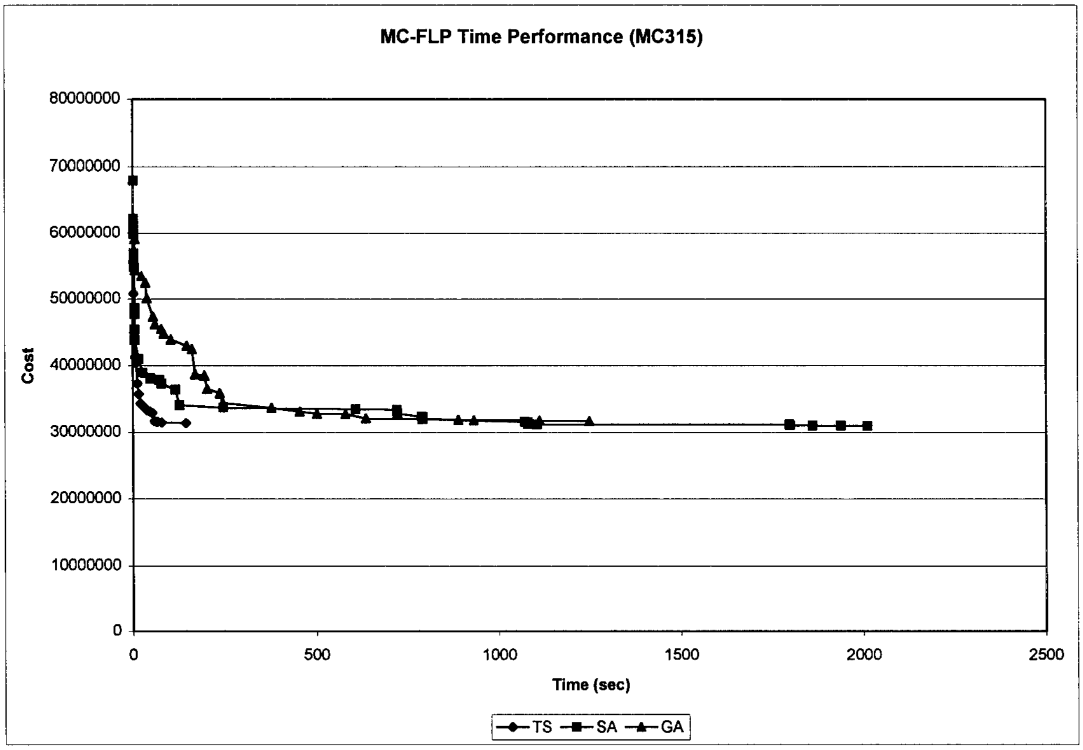

Figure 294. MC-FLP Time Performance (MC315)

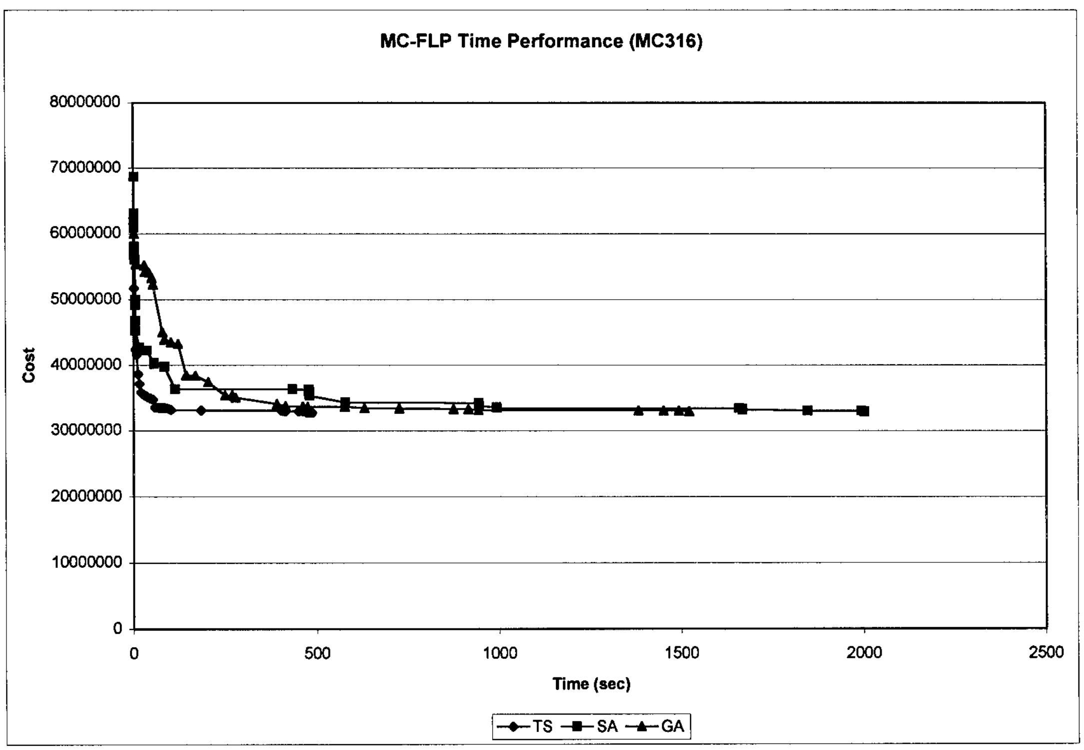

Figure 295. MC-FLP Time Performance (MC316)

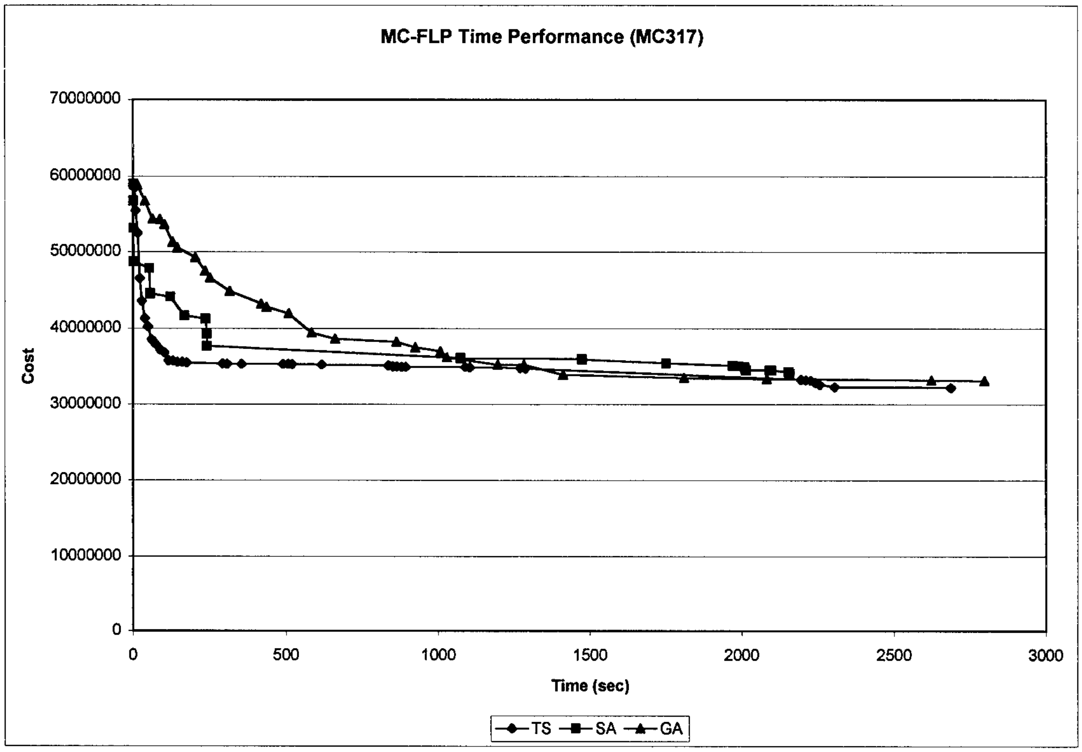

Figure 296. MC-FLP Time Performance (MC317)

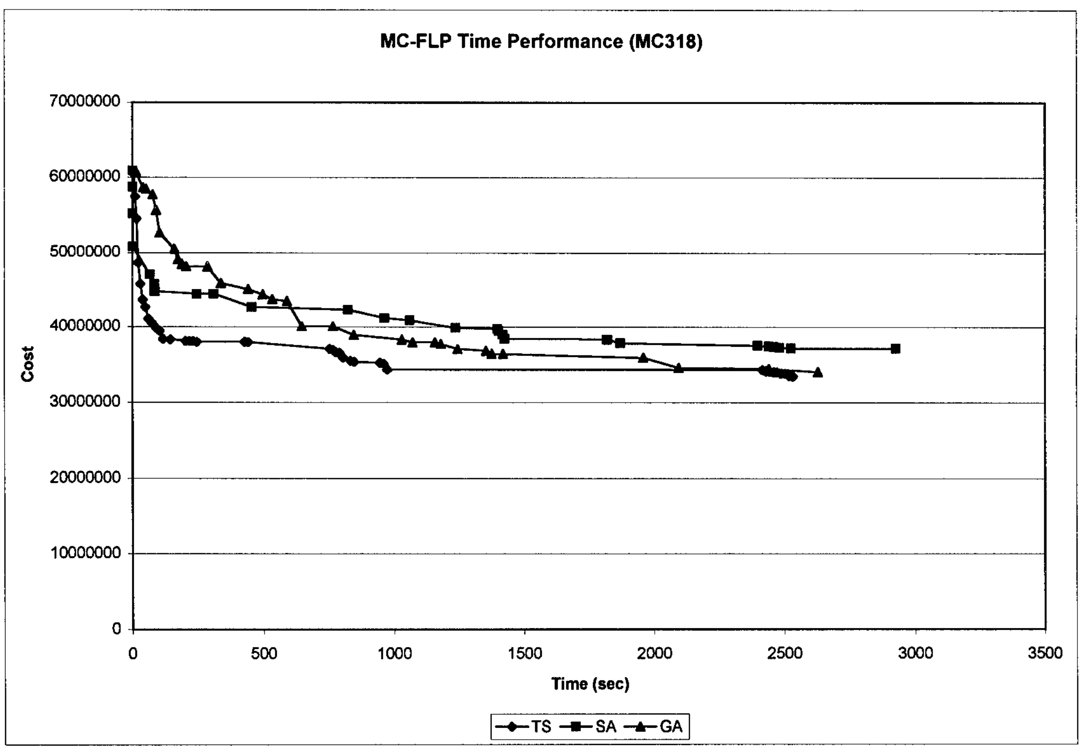

Figure 297. MC-FLP Time Performance (MC318)

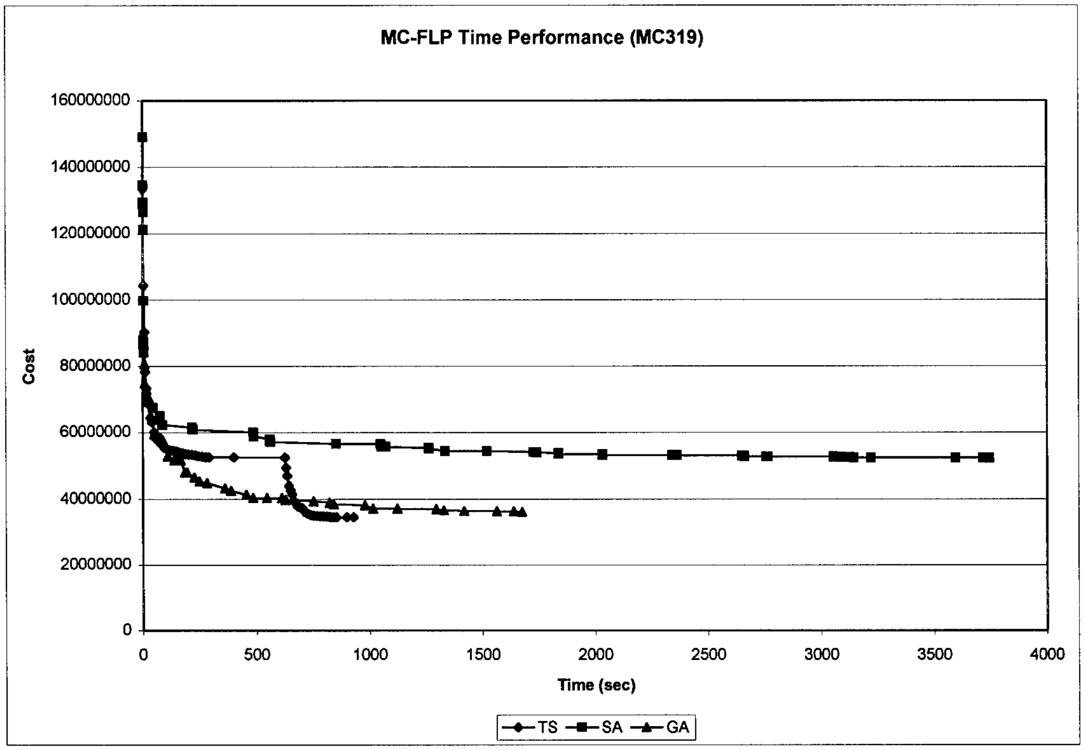

Figure 298. MC-FLP Time Performance (MC319)

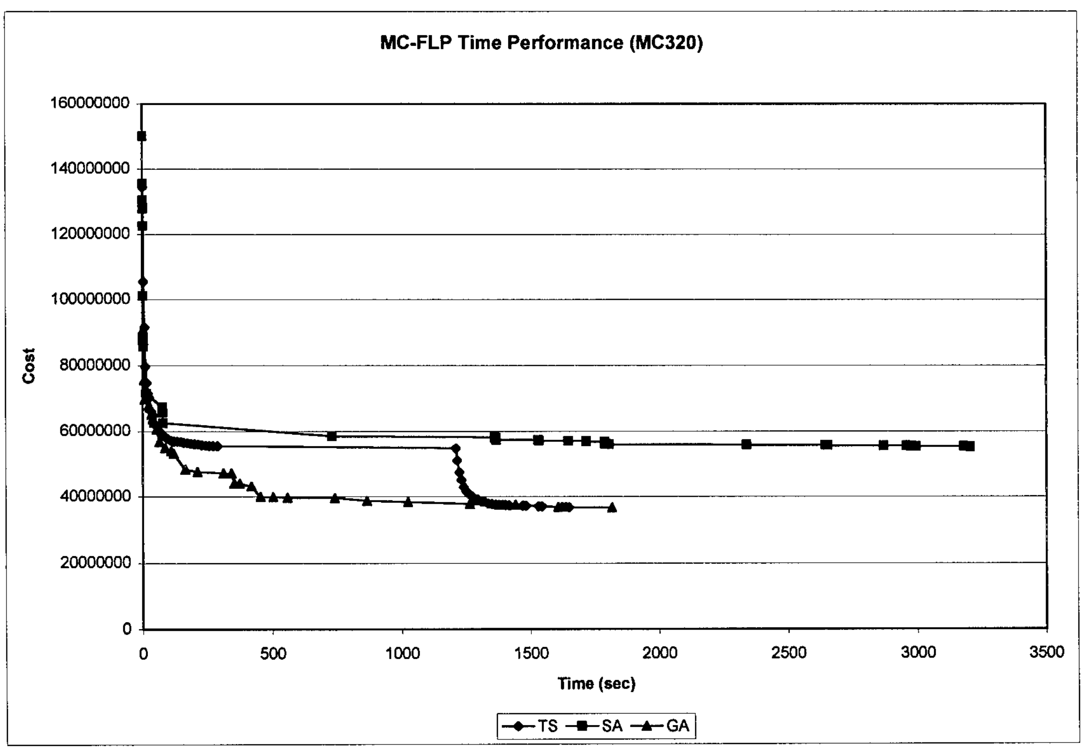

Figure 299. MC-FLP Time Performance (MC320)

APPENDIX J

MC-FLP SOLUTIONS PERFORMANCE CHARTS

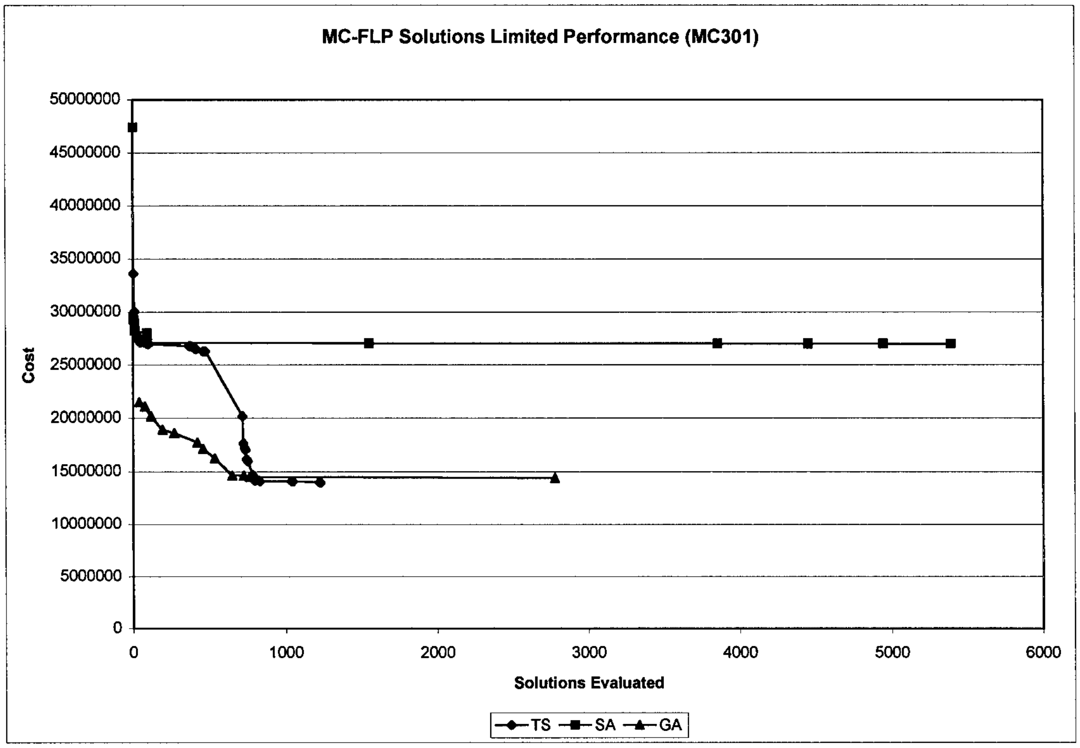

Figure 300. MC-FLP Solutions Performance (MC301)

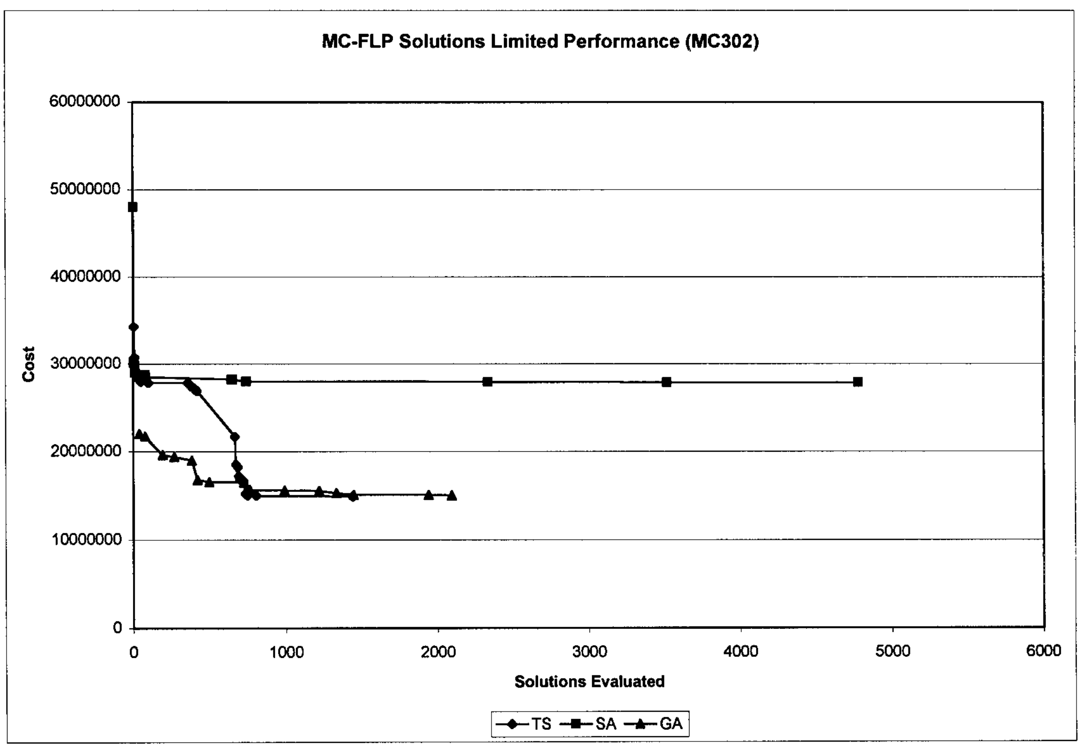

Figure 301. MC-FLP Solutions Performance (MC302)

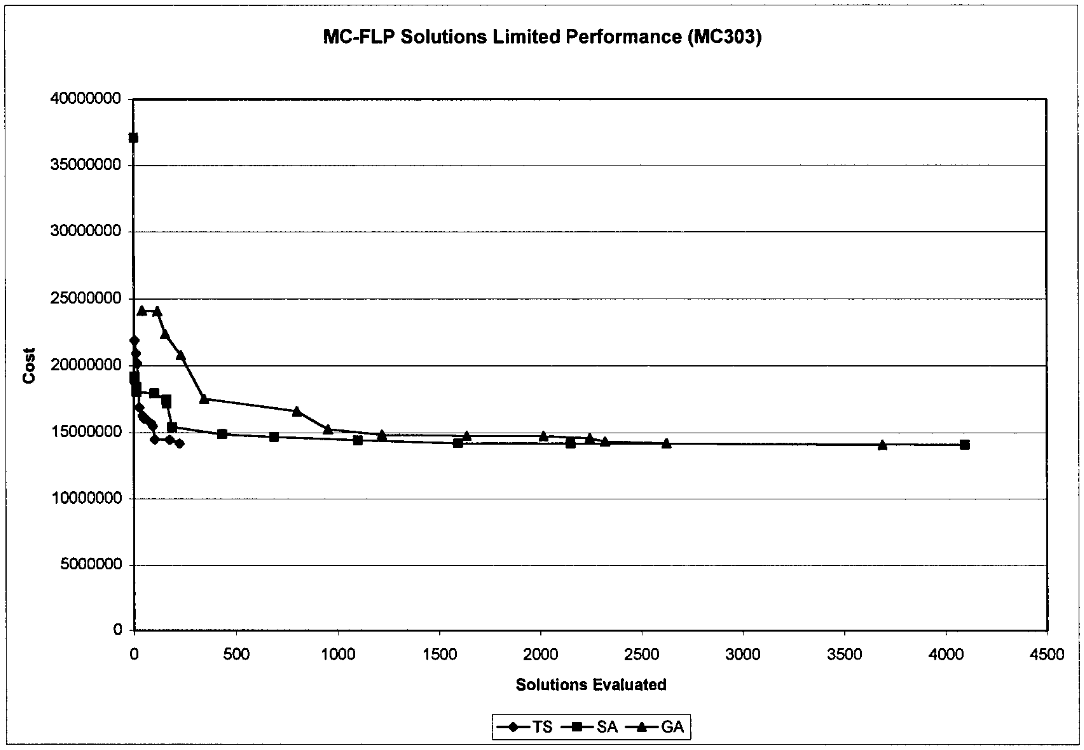

Figure 302. MC-FLP Solutions Performance (MC303)

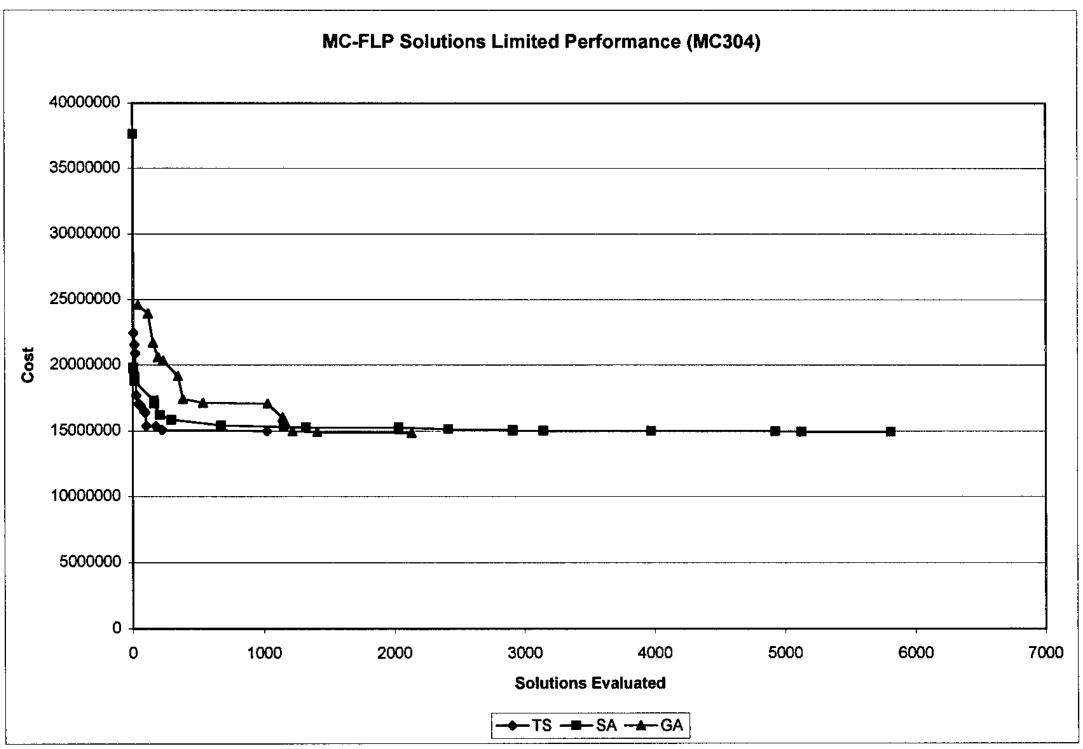

Figure 303. MC-FLP Solutions Performance (MC304)

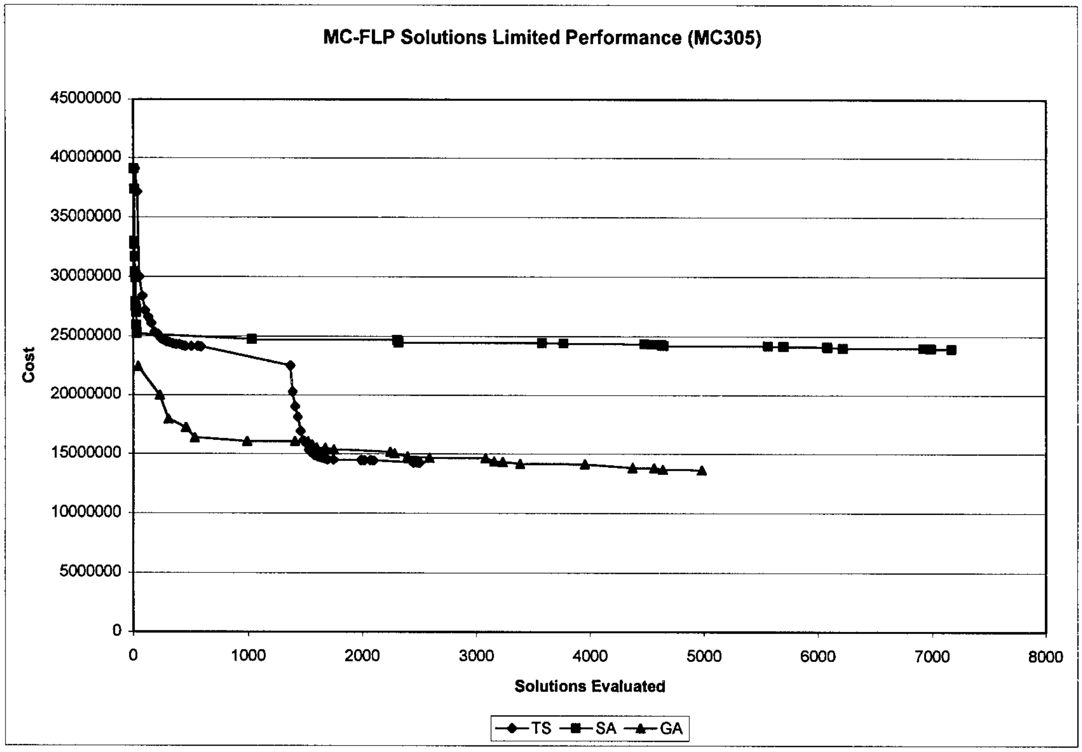

Figure 304. MC-FLP Solutions Performance (MC305)

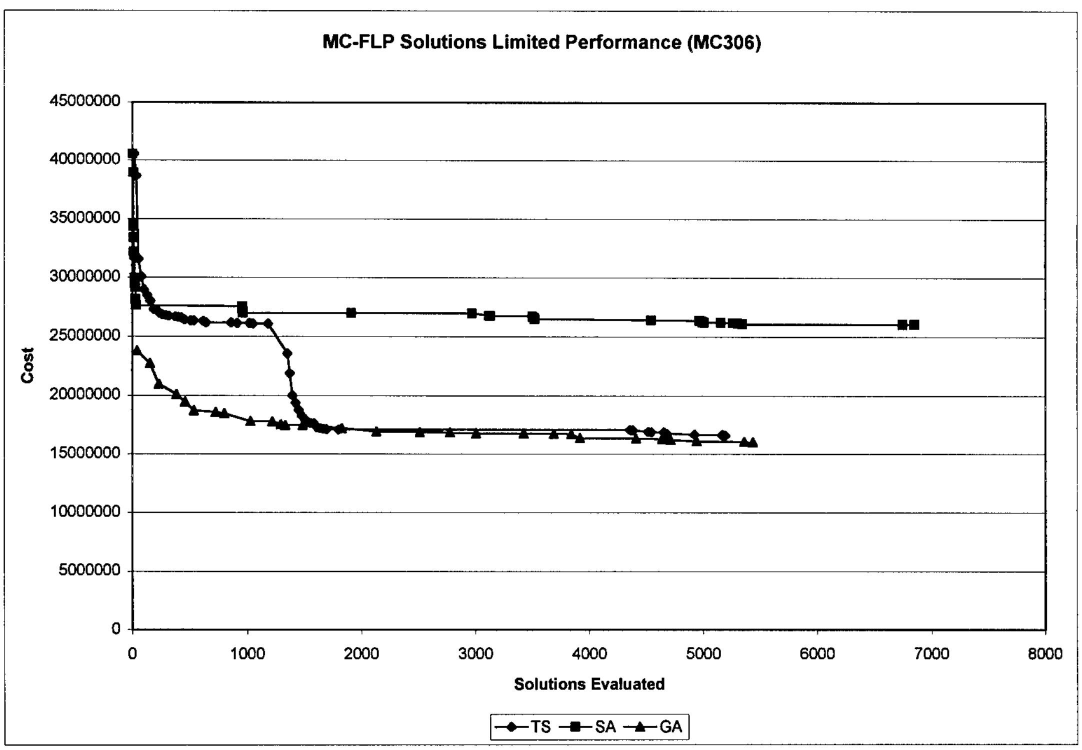

Figure 305. MC-FLP Solutions Performance (MC306)

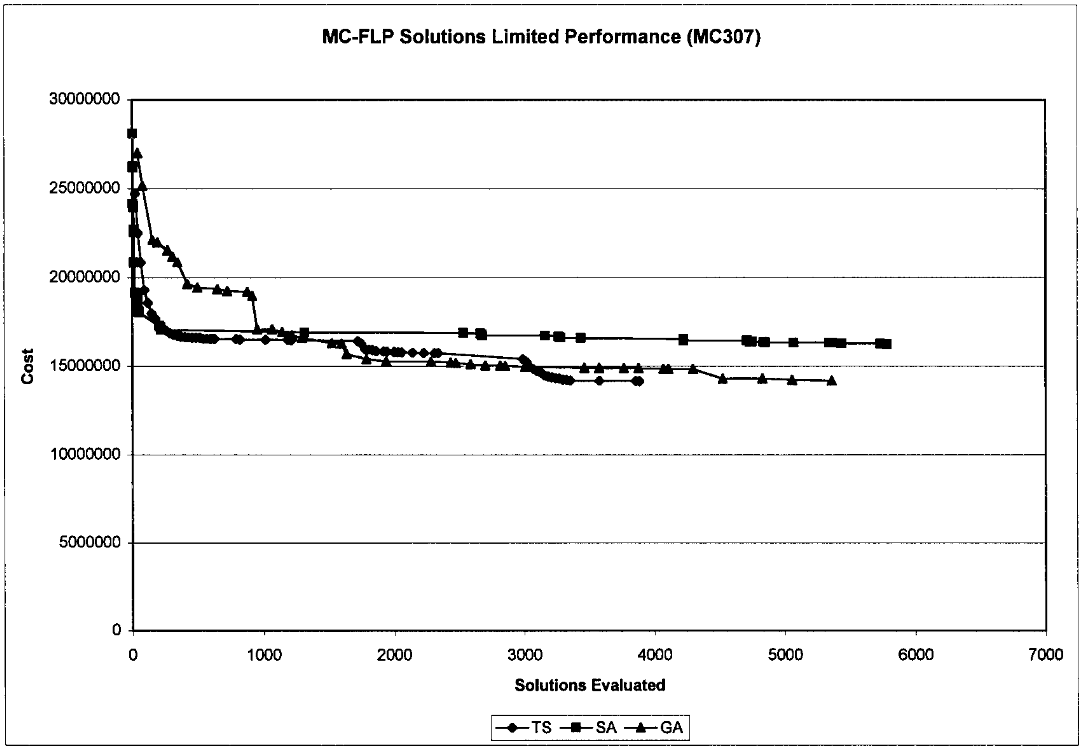

Figure 306. MC-FLP Solutions Performance (MC307)

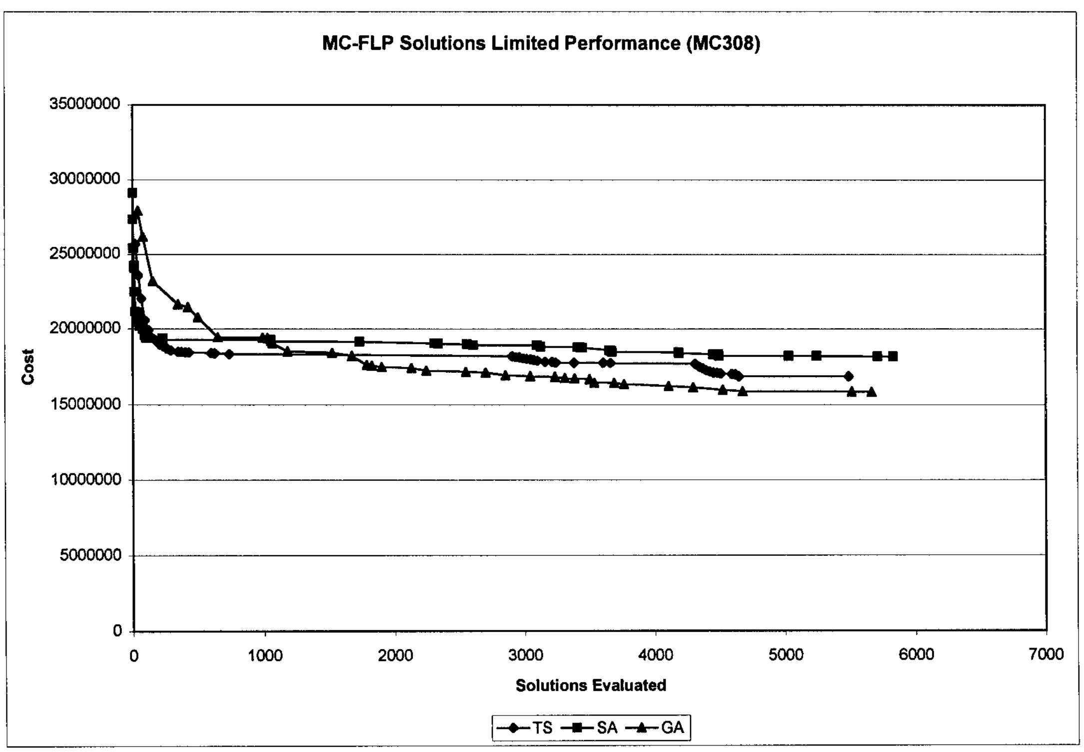

Figure 307. MC-FLP Solutions Performance (MC308)

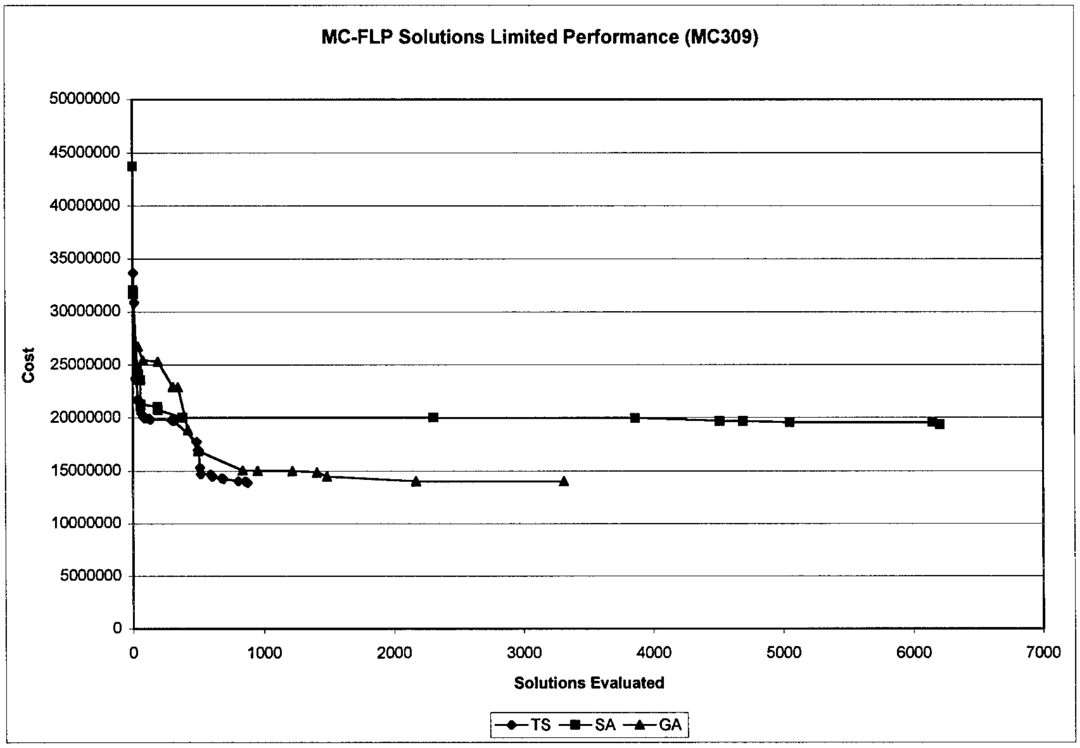

Figure 308. MC-FLP Solutions Performance (MC309)

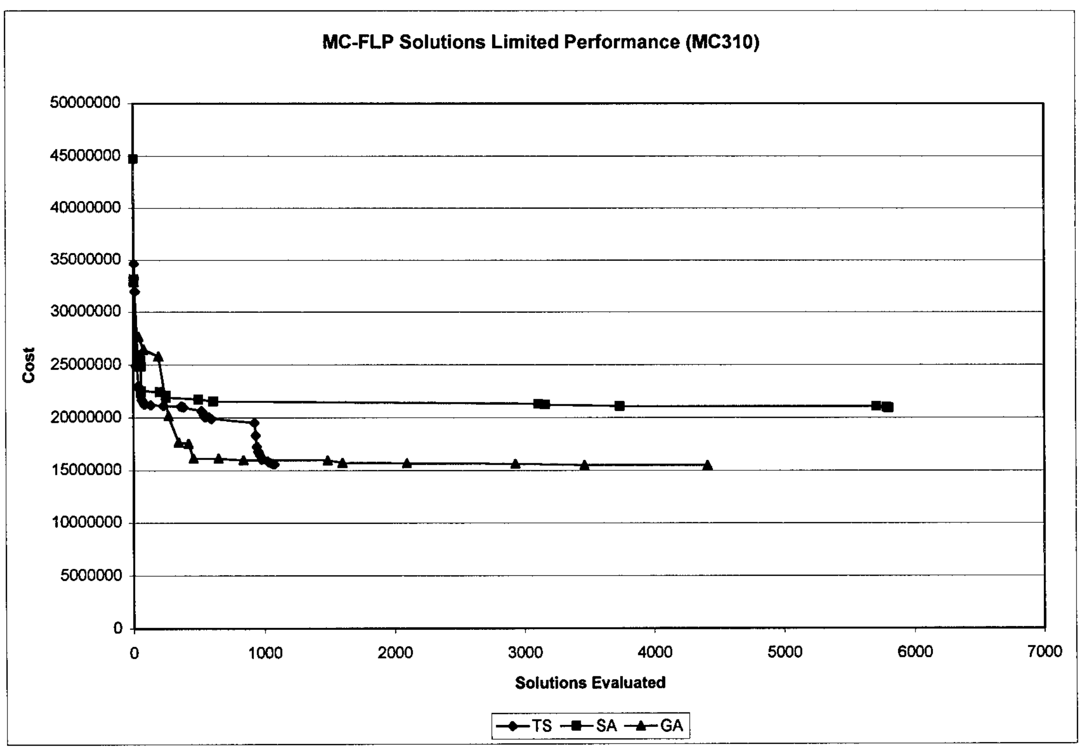

Figure 309. MC-FLP Solutions Performance (MC310)

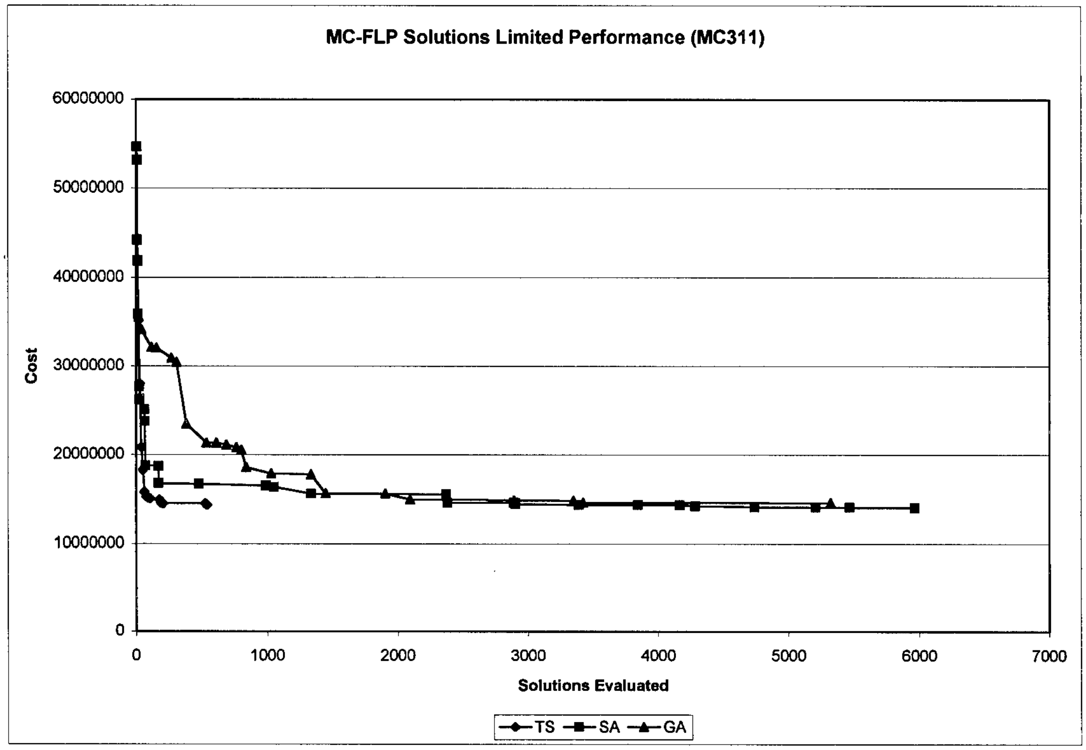

Figure 310. MC-FLP Solutions Performance (MC311)

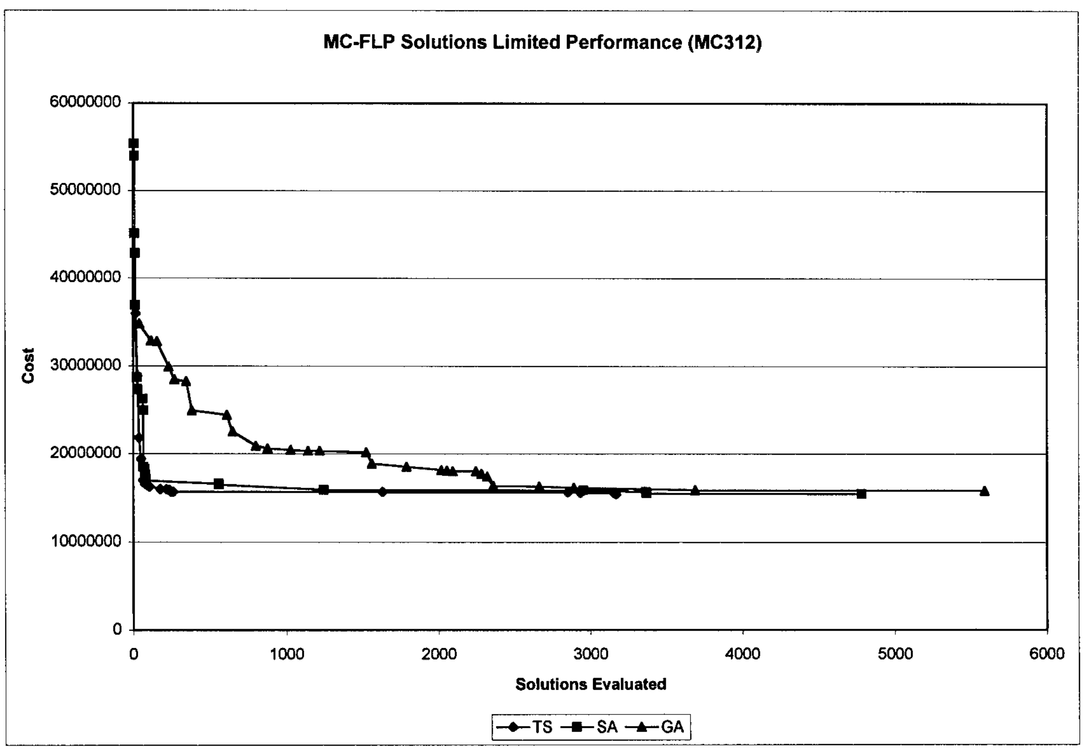

Figure 311. MC-FLP Solutions Performance (MC312)

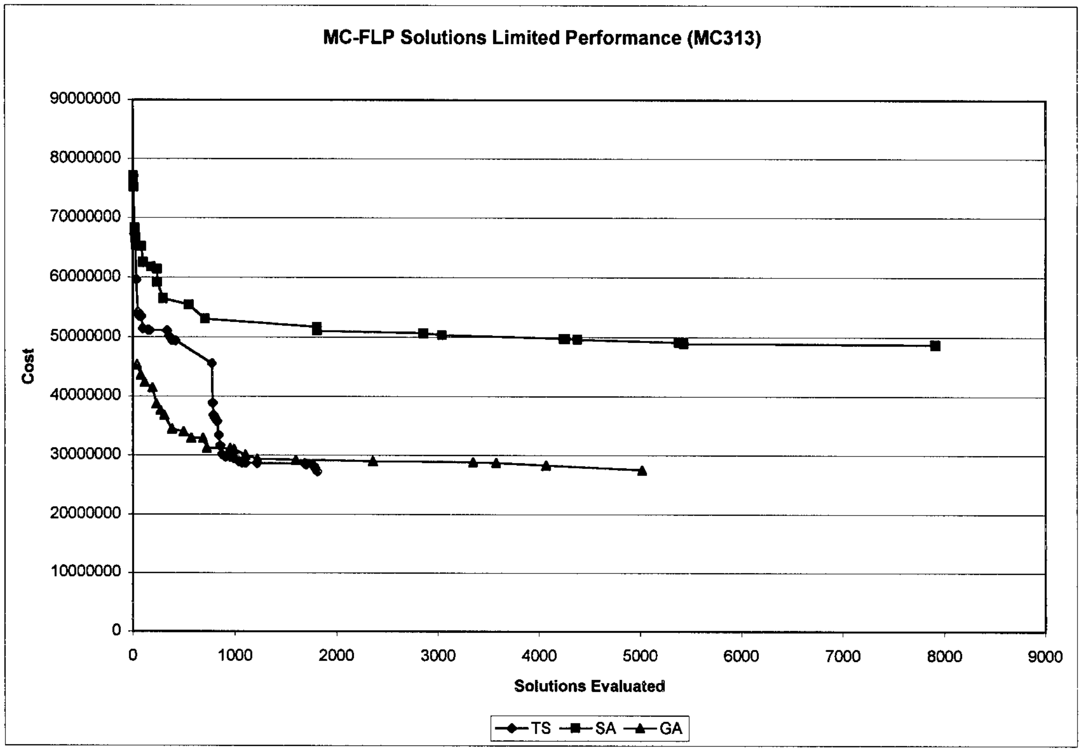

Figure 312. MC-FLP Solutions Performance (MC313)

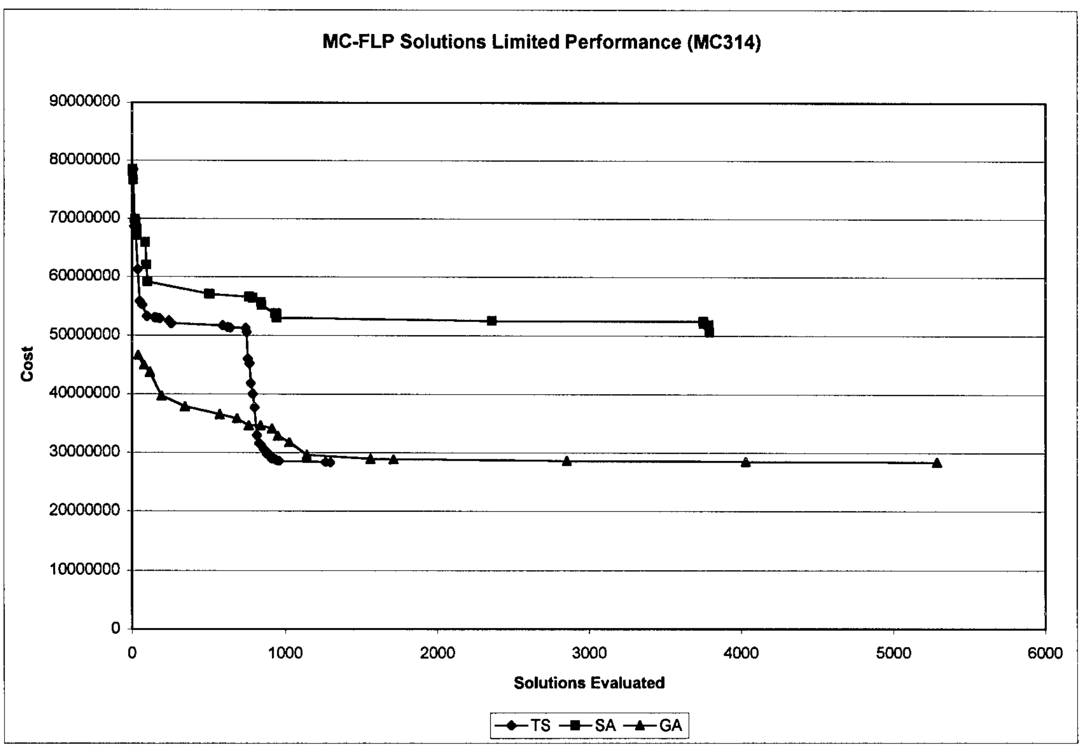

Figure 313. MC-FLP Solutions Performance (MC314)

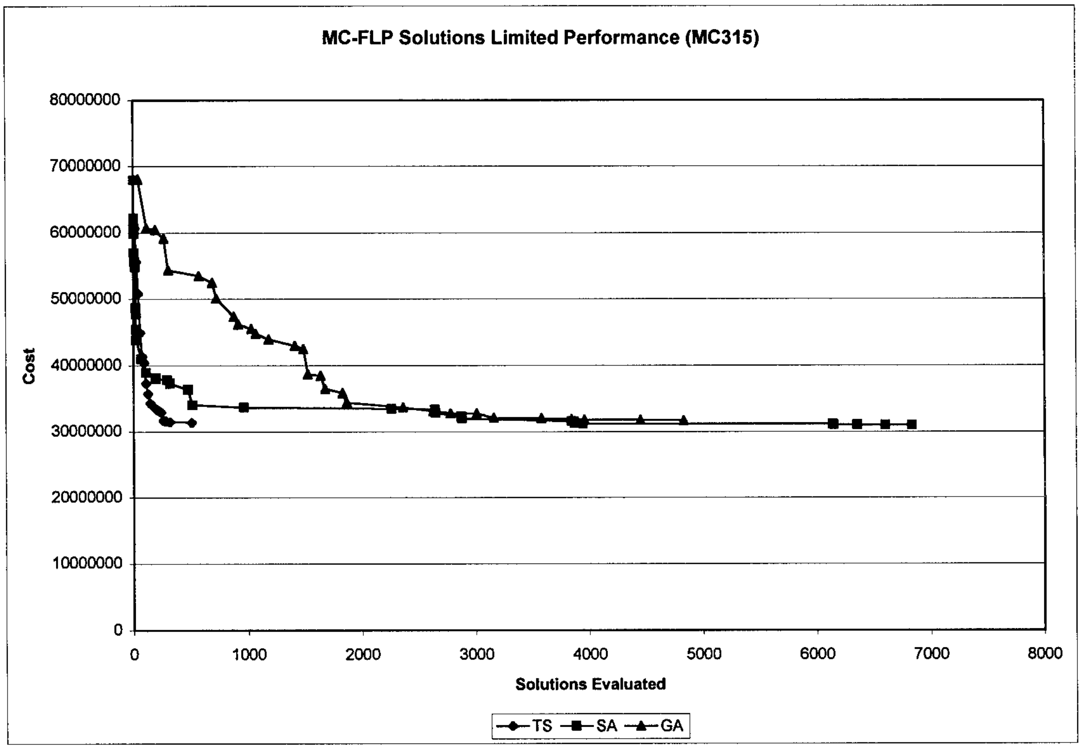

Figure 314. MC-FLP Solutions Performance (MC315)

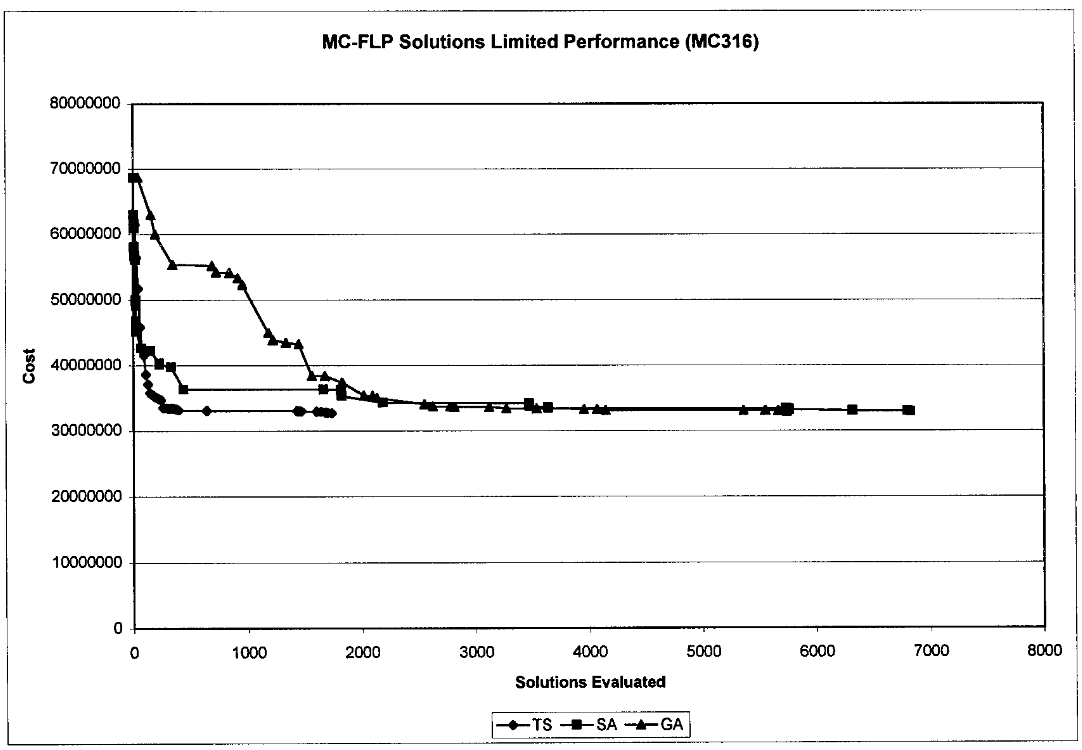

Figure 315. MC-FLP Solutions Performance (MC316)

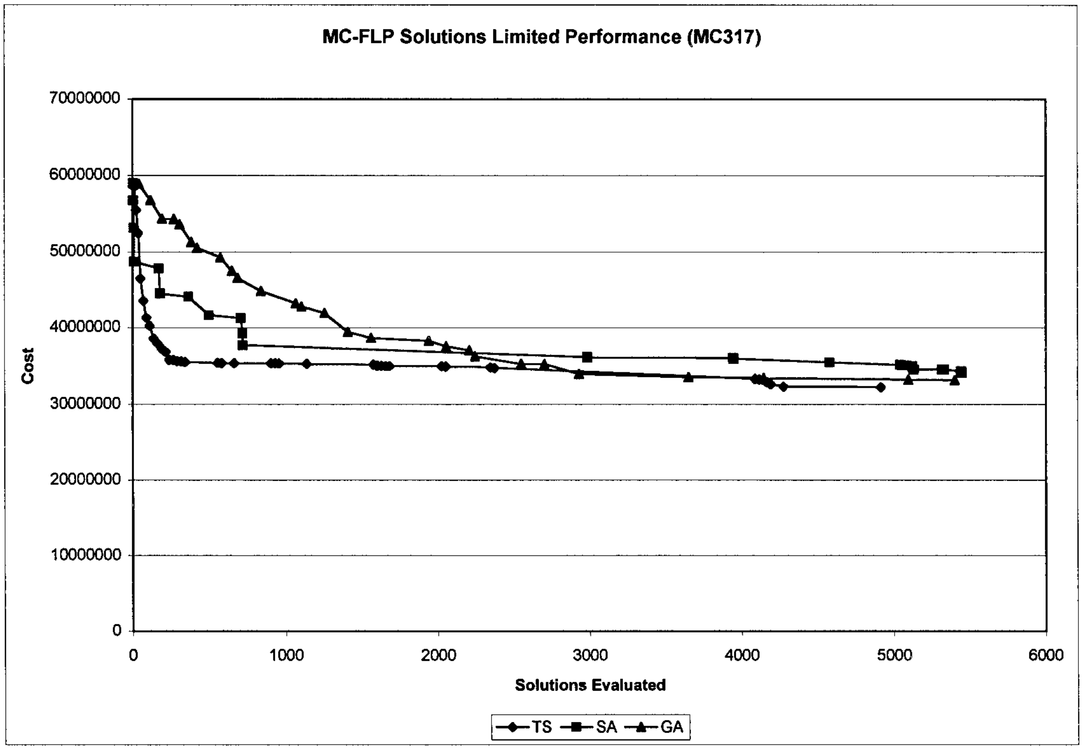

Figure 316. MC-FLP Solutions Performance (MC317)

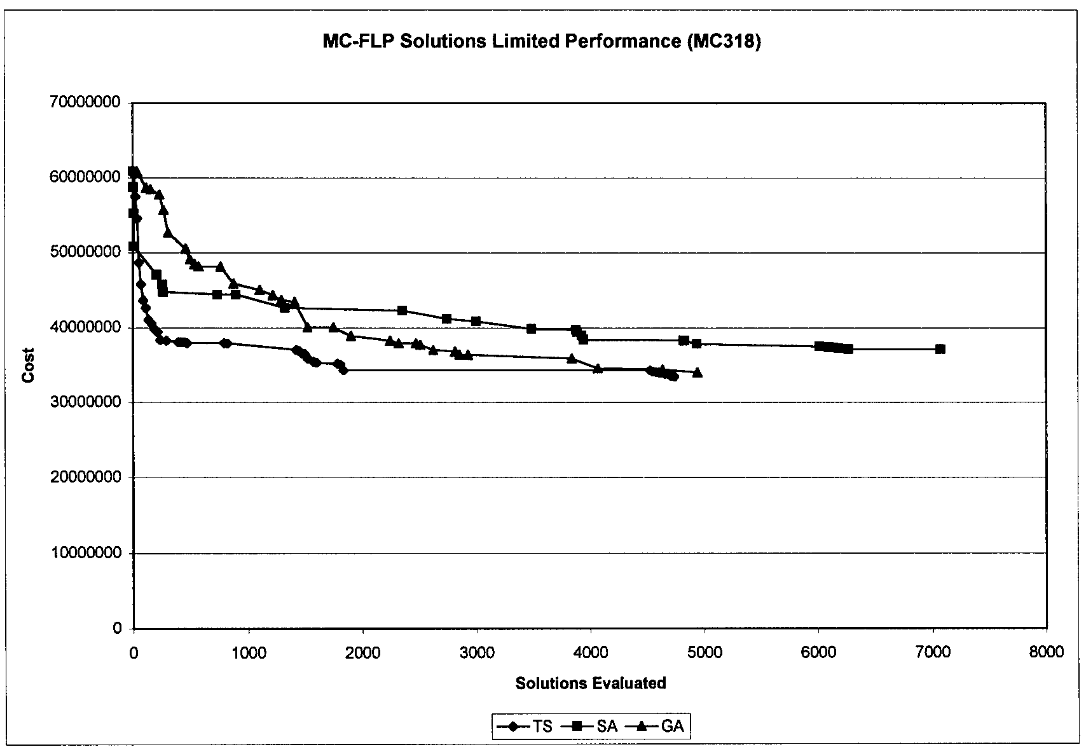

Figure 317. MC-FLP Solutions Performance (MC318)

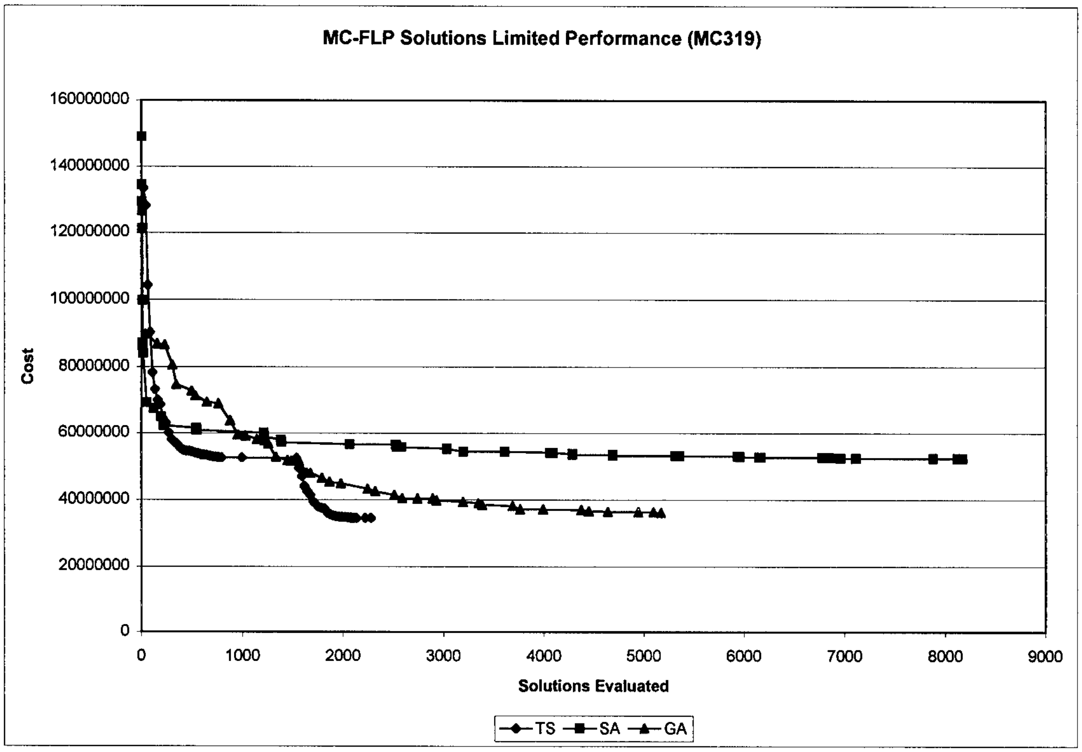

Figure 318. MC-FLP Solutions Performance (MC319)

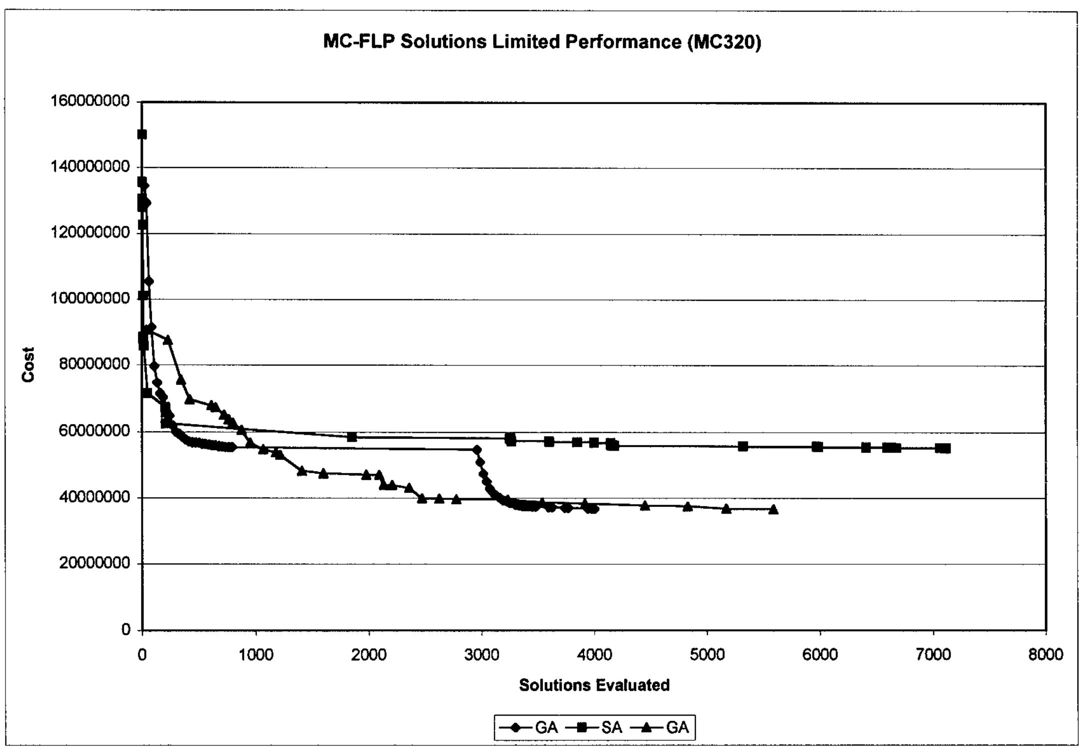

Figure 319. MC-FLP Solutions Performance (MC320)

APPENDIX K

COMPUTER SOFTWARE

For the purposes of this dissertation, twelve computer programs were developed. The following is a list of the nine heuristic programs and three optimal programs:

TSCAP	Tabu Search for CFLP
SACAP	Simulated Annealing for CFLP
GACAP	Genetic Algorithm for CFLP
TSMP	Tabu Search for MP-FLP
SAMP	Simulated Annealing for MP-FLP
GAMP	Genetic Algorithm for MP-FLP
TSMC	Tabu Search for MC-FLP
SAMC	Simulated Annealing for MC-FLP
GAMC	Genetic Algorithm for MC-FLP
BBCAP	Branch and Bound for CFLP (Optimal)
DPMP	Dynamic Programming for MP-FLP (Optimal)
BBMC	Branch and Bound for MC-FLP (Optimal)

All of the programs listed above were written in Microsoft's Visual C++ (v1.52). All of the programs are functional Windows software and run under Window 3.1 and Windows 95. Unfortunately, due to the large size of each

program (approximately 60 printed pages each), it was not feasible to include the source code as appendices. However, all materials are readily available from the author including the source code, executables, and test cases files. The author may be contacted via e-mail at marostegui@worldnet.att.net.

REFERENCES

Adenso-Díaz, Belarmino (1996) An SA/TS Mixture Algorithm for the Scheduling Tardiness Problem. *European Journal of Operational Research*, **88**, 516-524.

Akinc, Umit (1973) A Branch And Bound Procedure for Solving Warehouse Location Problems with Capacity Constraints. *Unpublished Ph.D. Dissertation,* University of Houston.

Akinc, Umit and B.M. Khumawala (1977) An Efficient Branch and Bound Algorithm for the Capacitated Warehouse Location Problem. *Management Science*, **23**(6), 585-594.

Balakrishnan, P. V. and Varghese S. Jacob (1996) Genetic Algorithms for Product Design. *Management Science*, **42**(8), 1105-1117.

Barnes, J. Wesley and Manuel Laguna (1993) Solving the Multiple-Machine Weighted Flow Time Problem Using Tabu Search. *IIE Transactions,* 25(2), 121-128.

Barnes, J. Wesley and John B. Chambers (1995) Solving the Job Shop Scheduling Problem with Tabu Search. *IIE Transactions,* **27,** 257-263.

Ben-Daya, M., and M. Al-Fawzan (1996) A Simulated Annealing Appraoch for the One-Machine Mean Tardiness Scheduling Problem. *European Journal of Operational Research,* **93**, 61-67.

Bulgak, A. A., P. D. Diwan, and B. Inozu (1995) Buffer Size Optimization in Asynchronous Assembly Systems Using Genetic Algorithms. *Computers and Industrial Engineering,* **28**(2), 309-322.

Černý, V. (1985) Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications,* **45**, 41-51.

Chan, K. C. and H. Tansri (1994) A Study of Genetic Crossover Operations on the Facilities Layout Problem. *Computers and Industrial Engineering,* **26**(3), 537-550.

Cheh, K. M., J. B. Goldberg and R. G. Askin (1991) A Note on the Effect of Neighborhood Structure in Simulated Annealing. *Computers and Operations Research*, **18,** 537-547.

Chen, W.-H., and B. Srivastava (1994) Simulated Annealing Procedures for Forming Machine Cells in Group Technology. *European Journal of Operational Research,* **75,** 100-111.

Cheng, Runwei, Mitsuo Gen, and Yasuhiro Tsujimura (1996) A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms—I. Representation. *Computers and Industrial Engineering,* **30**(4), 983-997.

Collins, N. E., R. W. Eglese and B. L. Golden (1988) Simulated Annealing—An Annotated Bibliography. *American Journal of Mathematical Management Science,* **3**(3), 209-307.

Connolly D. (1992) General Purpose Simulated Annealing. *Journal of the Operational Research Society,* **43,** 495-505.

Crainic, Teodor G., Michel Gendreau, Patrick Soriano, and Michel Toulouse (1993) A Tabu Search Procedure for Multi-commodity Location/Allocation with Balancing Requirements. *Annals of Operations Research,* **41,** 359-383.

Davis P.S. and T.L. Ray (1969) A Branch-Bound Algorithm for the Capacitated Facilities Location Problem. *Naval Research Logistics Quarterly.* **16**, 331-344.

Dowsland, Kathryn A. (1996) Genetic Algorithms—a Tool for OR? *Journal of the Operational Research Society,* **47,** 550-561.

Dowsland, Kathryn A. (1993) Some Experiments with Simulated Annealing Techniques for Packing Problems. *European Journal of Operational Research,* **68,** 389-399.

Dudewicz, E.J. and S.R. Dalal (1975) Allocation of Observations in Ranking and Selection with unequal Variances. *Sankhya,* **B37**, 28-78.

Eglese, R. W. (1990) Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research,* **46,** 271-281.

Ellwein, L.B. and P. Gray (1971) Solving Fixed Charge Location-Allocation Problems with Capacity and Configuration Constraints. *AIIE Transactions,* **3,** 290-299.

França, Paulo M., Michel Gendreau, Gilbert Laporte, and Felipe M. Müller (1996) A Tabu Search Heuristic for the Multiprocessor Scheduling Problem With Sequence Dependent Setup Times. *International Journal of Production Economics*, **43**, 79-89.

Garavelli, A.C., O.G. Okogbaa, and V. Nicola (1996) Global Manufacturing Systems: A Model Supported by Genetic Algorithms to Optimize Production Planning. *Computers & Industrial Engineering,* **31**, 193-196.

Gendreau, Michel, Alain Hertz, and Gilbert Laporte (1994) A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science,* **40**(10), 1276-1290.

Geoffrion, A.M. and G.W. Graves (1974) Multi-commodity Distribution Systems Design. *Management Science*, **20**, 822-844.

Glover, F. (1977) Heuristic for Integer Programming Using Surrogate Constraints. *Decision Sciences,* **8,** 156-166.

Glover, F. (1989) Tabu search—Part I. *ORSA Journal on Computing,* **1**(3), 190-206.

Glover, F. (1990a) Tabu Search—Part II. *ORSA Journal on Computing,* **2**(1), 4-32.

Glover, F. (1990b) Tabu Search: A Tutorial. *Interfaces,* **20**(4), 74-94.

Glover, F. (1993) A User's Guide to Tabu Search. *Annals of Operations Research.*

Glover, Fred and Harvey J. Greenberg (1989) New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intlligence. *European Journal of Operational Research,* **39,** 119-130.

Goldberg, David E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Massachusetts: Addison Wesley Publishing Company, Inc.

Golden, B.L. and W.R. Stewart (1985) Empirical Analysis of Heuristics. In *The Travelling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., Chichester: John Wiley & Sons, 207-249.

Hajek, B. (1988) Cooling Schedules for Optimal Annealing. *Mathematics and Operations Research,* **13**(2), 311-329.

Hao, Qi, Zihou Yang, Dingwei Wang, and Zheng Li (1996) Common Due-Date Determination and Sequencing Using Tabu Search. *Computers and Operations Research*, **23**(5), 409-417.

He, Zesheng, Taeyong Yang, and Andy Tiger (1996) An Exchange Heuristic Imbedded with Simulated Annealing for Due-Dates Job-Shop Scheduling. *European Journal of Operational Research,* **91**, 99-117.

Hindi, K. S. (1995) Solving the Single-Item, Capacitated Dynamic Lot-Sizing Problem with Startup and Reservation Costs by Tabu Search. *Computers and Industrial Engineering,* **28**(4), 701-707.

Holland, John H. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor: The university of Michigan Press.

Hormozi, A.M. (1987) An Improved Multi-Period Facility Location Problem. *Unpublished Ph.D. Dissertation*, University of Houston.

Hormozi, A.M. and B.M. Khumawala (1996) An Improved Multi-Period Facility Location Problem. *IIE Transactions*, **28**, 105-114.

Houck, Christopher R., Jeffrey A. Joines and Michael G. Kay (1996) Comparison of Genetic Algorithms, Random Restart and Two-Opt Switching for Solving Large Location-Allocation Problems. *Computers and Operations Research*, **23**(6) 587-596.

Icmeli, Oya and S. Selcuk Erenguc (1994) A Tabu Search Procedure for the Resource Constrained Project Scheduling Problem With Discounted Cash Flows. *Computers and Operations Research*, **21**(8), 841-853.

Ishibuchi, Hisao, Shinta Misaki, and Hideo Tanaka (1995) Modified Simulated Annealing Algorithms for the Flow Show Sequencing Problem. *European Journal of Operational Research,* **81,** 388-398.

Joines, Jeffrey A., C. Thomas Culbreth, and Russell E. King (1996) Manufacturing Cell Design: An Integer Programming Model Employing Genetic Algorithms. *IIE Transactions,* **28**, 69-85.

Khumawala, B.M. (1974) An Efficient Heuristic Procedure for the Capacitated Warehouse Location Problem. *Naval Research Logistics Quarterly,* **21(**4), 609-623.

Kim, Jung-Ug and Yeong-Dae Kim (1996) Simulated Annealing and Genetic Algorithms for Scheduling Products With Multi-Level Product Structure. *Computers and Operations Research*, **23**(9), 857-868.

Kim, Yeo Keun, Yong Ju Kim, and Yeongho Kim (1996) Genetic Algorithms for Assembly Line Balancing With Various Objectives. *Computers and Industrial Engineering,* **30**(3), 397-409.

Kim, Yeong-Dae, Hyeong-Gyu Lim, and Moon_Won Park (1996) Search Heuristics for a Flowshop Scheduling Problem in a Printed Circuit Board Assembly Process. *European Journal of Operational Research,* **91**, 124-143.

Kincaid, Rex K. (1992) Good Solutions to Discrete Noxious Location Problems Via Metaheuristics. *Annals of Operations Research,* **40,** 265-281.

Kirkpatrick S., C. Gelatt, and P. Vecchi (1983) Optimization by Simulated Annealing. *Science,* **220,** 671-679.

Koulamas, C., S. R. Antony, and R. Jaen (1994) A Survey of Simulated Annealing Applications to Operations Research Problems. *Omega, International Journal of Management Science,* **22**(1), 41-56.

Kuehn Alfred A. and Michael J. Hamburger (1963) A Heuristic Program for Locating Warehouses. *Management Science,* **9**(4), 643-666.

Kuik, Roelof, Marc Salomon, Luk N. Van Wassenhove, and Johan Maes (1993) Linear Programming, Simulated Annealing and Tabu Search Heuristics for Lotsizing in Bottleneck Assembly Systems. *IIE Transactions,* 25(1), 62-72.

Laguna, Manuel, James P. Kelly, Jose Luis Gonzalez-Velarde, and Fred Glover (1995) Tabu Search for the Multilevel Generalized Assignment Problem. *European Journal of Operational Research,* **82,** 176-189.

Law, Averill M. and W. David Kelton (1991) *Simulation Modeling & Analysis,* 2nd ed., McGraw-Hill, Inc.

Lee, Jae-Kwan and Yeong-Dae Kim (1996) Search Heuristics for Resource Constrained Project Scheduling. *Journal of the Operational Research Society,* **47,** 678-689.

Leu Yow-Yuh, Lance A. Matheson, and Loren Paul Rees (1994) Assembly Line Balancing Using Genetic Algorithms With Heuristic-Generated Initial Populations and Multiple Evaluation Criteria. *Decision Sciences,* 25(4), 581-606.

Leu, Yow-Yuh, Lance A. Matheson, and Loren Paul Rees (1996) Sequencing Mixed-Model Assembly Lines With Genetic Algorithms. *Computers and Industrial Engineering,* **30**(4), 1027-1036.

Liu, Chih-Ming, Ruey-Li Kao, and An-Hsiang Wang (1994) Solving Location-Allocation Problems with Rectilinear Distances by Simulated Annealing. *Journal of the Operational Research Society,* **45**(11), 1304-1315.

Lundy, M. and A. Mees (1986) Convergence of an Annealing Algorithm. *Mathematical Programming,* **34,** 111-124.

Malmborg, Charles J. (1996) A Genetic Algorithm for Service Level Based Vehicle Scheduling. *European Journal of Operational Research,* **93**, 191-134.

Marett, Richard and Mike Wright (1996) A Comparison of Neighborhood Search Techniques for Multi-Objective Combinatorial Problems. *Computers and Operations Research*, **23**(5), 465-483.

Metropolis, N. A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953) Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics,* **21,** 1087-1091.

Murata, Tadahiko, Hisao Ishibuchi, and Hideo Tanaka (1996) Genetic Algorithms for Flowshop Scheduling Problems. *Computers and Industrial Engineering,* **30**(4), 1061-1071.

Neebe, A.W. and B.M. Khumawala (1981) An Improved Algorithm for the Multi-Commodity Location Problem. *Journal of Operational Research Society,* **32**, 143-149.

Nowicki, Eugeniusz, and Czeslaw Smutnicki (1996) A Fast Tabu Search Algorithm for the Permutation Flow Shop Problem. *European Journal of Operational Research,* **91**, 160-175.

Nowicki, E. and S. Zdrazalka (1996) Single Machine Scheduling With Major and Minor Setup Times: A Tabu Search Approach. *Journal of the Operational Research Society,* **47**, 1054-1064.

Osman, I. H. and C. N. Potts (1989) Simulated Annealing for Permutation Flow-Shop Scheduling. *Omega, International Journal of Management Science,* **17**(6), 551-557.

Reeves, Colin R., ed. (1993) *Modern Heuristic Techniques for Combinatorial Problems.* New York: John Wiley & Sons.

Renaud, Jacques, Gilbert Laporte, and Gayez F. Boctor (1996) A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem. *Computers and Operations Research*, **23**(3), 229-235.

Roach, Anthony and Rakesh Nagi (1996) A Hybrid GA-SA Algorithm for Just-In-Time Scheduling of Multi-Level Assemblies. *Computers and Industrial Engineering*, **30**(4), 1047-1060.

Rolland, Erik, David A. Schilling, and John R. Current (1996) An Efficient Tabu Search Procedure for the p-Median Problem. *European Journal of Operational Research,* **96,** 329-342.

Roodman, G.M. and L.B. Schwarz (1975) Optimal and Heuristic Facility Phase-out Strategies. *AIEE Transactions*, **7**, 177-184.

Roodman, G.M. and L.B. Schwarz (1977) Extension of the Multi-Period Facility Phase-Out Model; New Procedures and Applications to a Phase-in/Phase-out Problem. *AIEE Transactions*, **9**, 103-107.

Sá., G. (1969) Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem. *Operations Research,* **17**, 1005-1016.

Sakawa, Masatoshi, Kosuke Kato, and Tetsuya Mori (1996) Flexible Scheduling in a Machining Center Through Genetic Algorithms. *Computers and Industrial Engineering,* **30**(4), 931-940.

Schmidt, Linda C and John Jackman (1995) Evaluating Assembly Sequences for Automatic Assembly Systems. *IIE Transactions,* **27,** 23-31.

Sinclair, Marius (1993) Comparison of the Performance of Modern Heuristics for Combinatorial Optimization on Real Data. *Computers and Operations Research*, **20**(7), 687-695.

Skorin-Kapov, Darko and Jadranka Skorin-Kapov (1994) On Tabu Search for the Location of Interacting Hub Facilities. *European Journal of Operational Research,* **73,** 502-509.

Sweeney, D.J. and Ronald L. Tatham (1976) An Improved Long Run Model for Multiple Warehouse Location. *Management Science*, **22**(7), 748-758.

Taillard, E. (1991) Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing,* **17,** 433-455.

Taillard, Éric D., Gilbert Laporte, and Michel Gendreau (1996) Vehicle Routeing with Multiple Use of Vehicles. *Journal of the Operational Research Society,* **47**, 1065-1070.

Thompson, Gary M. (1996) A Simulated Annealing Heuristic for Shift Scheduling Using Non-Continuously Available Employees. *Computers and Operations Research*, **23**(3), 275-288.

Van Laarhoven, Peter J. M., and E. H. L. Aarts (1987) *Simulated Annealing: Theory and Applications.* Reidel, Dordrecht.

Van Laarhoven, Peter J. M., Emile H. L. Aarts, and Jan Karel Lenstra (1992) Job Shop Scheduling by Simulated Annealing. *Operations Research,* **40**(1), 113-125.

Vignaux, G. A., and Z. Michalewicz (1991) A Genetic Algorithm for the Linear Transportation Problem. *IEEE Transactions on Systems, Man, and Cybernetics,* **21**(2), 445-452.

Warszawski, A. (1987) Multi-Dimensional Location Problems. *Operational Research Quarterly*, **24**, 165-179.

Wilhelm, Mickey R. and Thomas L. Ward (1987) Solving Quadratic Assignment Problems by Simulated Annealing. *IIE Transactions*, **107**, 107-119.